Predictive Analytics for Sales Forecasting

Master of Science in **Data Science**

Submitted By:

Name: Priyanshu Sing

ABSTRACT

This project is a comprehensive implementation of predictive analytics solutions for sales forecasting for the financial and insurance verticals This study is called Predictive Analytics for Sales Forecasting in the Finance Sector, and the data set it uses includes historical information from insurance agencies from 2005 to 2014 (213,328 records from 1,623 insurance agencies in 6 states of the United States).

The goal of this project is to build and test predictive models which would be able to predict the written premium accurately from past financial metrics such as loss ratio, retention ratio, new business premium, and incurred losses The methods that were employed are Linear Regression (baseline), Random Forest Regressor, and Facebook Prophet for time-series forecasting, all three of which were implemented and compared.

This project uses Python as the main programming language, and some of the libraries utilized are Pandas, NumPy, Scikit-learn, and Prophet for data processing and machine learning respectively Structured data was queried using SQL (SQLite) whereas Power BI was used to create an interactive 3-page dashboard, offering a descriptive and predictive analytical insight.

The best results were obtained by the Random Forest Regressor on log-transformed premium values with an R^2 score of 0 848, a Mean Absolute Error (MAE) of 0 505 and an RMSE of 0 858 The two best indicators of total premium volume are incurring losses and new business written premium, according to key findings Commercial lines grew 114% over the last decade, and Ohio made up 58% of the total written premium portfolio.

The project builds the existing knowledge in the field of predictive analytics in finance and offers valuable information for data driven recommendations to enhance the accuracy of sales forecasting in the insurance industry.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Full Form
ML	Machine Learning
EDA	Exploratory Data Analysis
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
R^2	R-Squared (Coefficient of Determination)
RF	Random Forest
LR	Linear Regression
CL	Commercial Lines
PL	Personal Lines
SQL	Structured Query Language
BI	Business Intelligence
DFD	Data Flow Diagram
UML	Unified Modelling Language
ER	Entity Relationship
API	Application Programming Interface
CSV	Comma-Separated Values
KPI	Key Performance Indicator
YoY	Year-over-Year
UI	User Interface

CHAPTER 1: INTRODUCTION

1.1 Background of the Project

The insurance and financial services sector create huge transactional data on a daily basis Insurance agencies monitor written premiums, policy renewals, claims, loss ratios and new business, as well as thousands of client accounts Despite the wealth of historical information available, a lot of financial organizations are still using manual approaches, gut instinct or cruder statistical models to forecast sales success in the future Predictive Analytics is the predictive modelling process to identify future events from past trends using statistical algorithms, machine learning models, and data mining techniques As it relates to insurance sales, predictive analytics can revolutionize the way that agencies allocate resources, target their sales goals and find markets where growth is going to occur before it does Being able to predict written premium volumes one to three years ahead gives insurance agencies an edge in the competitive arena The financial services industry - including banking, insurance, investment management and financial advisory - is one of the most data-intensive sectors in the global economy Recent studies indicate that companies with sophisticated forecasting methods using advanced analytics can outperform companies with traditional forecasting methods in terms of forecast accuracy and operational efficiency Predictive analytics is now possible for every organization, thanks to the rise of cloud computing, open-source machine learning libraries, and business intelligence platforms. This project is targeted specifically at the insurance sub-sector of the finance sector with a real-world dataset of agency performance measures covering 11 years (2005 - 2015) of data. The data includes 1,623 independent insurance agencies in six states in the United States, offering a variety of commercial lines (CL) and personal lines (PL) insurance products The project investigates how to use several machine learning approaches to predict written premium sales and compare the models' performances using conventional statistical measures. The global insurance industry processes hundreds of millions of policy transactions every year and manages assets worth several trillion dollars. Inside that

scale, premium forecasting sits at the centre of almost every significant business decision an insurer makes - setting agency sales targets, allocating marketing spend, planning staffing, structuring reinsurance coverage, and communicating financial guidance to investors and regulators. Get the forecast wrong and the consequences don't stay contained: overstaffed or understaffed teams, missed reinsurance thresholds, guidance that doesn't hold up, budgets pointed in the wrong direction. Forecast errors in insurance aren't abstract - they hit the bottom line directly.

Regional insurers face a version of this problem that large national carriers mostly don't. A regional insurer typically distributes through independent agents who also represent competing carriers. That means limited visibility into each agent's full book of business and no easy way to predict which agents will grow, which will plateau, and which will quietly shrink their premium volumes in the coming year. Regional portfolios also tend to be more exposed to local weather events and economic conditions - the kind of sudden fluctuations that historical trend analysis struggles to anticipate. The dataset used in this project sits squarely in this context: 1,623 independent agencies across six states in the American Midwest, exactly the kind of regional portfolio where forecasting is hardest and most consequential.

What's changed in the last decade is that the tools to tackle this problem have become genuinely accessible. A regional insurer ten years ago might have had one or two actuaries running spreadsheet models. Today, a single analyst with open-source Python libraries - Scikit-learn, Prophet, Pandas - can build, train, and evaluate multiple predictive models on hundreds of thousands of records in a few hours, at close to zero incremental cost. You don't need a large technology budget or a dedicated data science team to implement machine learning-based forecasting anymore. That shift - advanced analytics moving from exclusive to accessible - is the primary motivation behind this project.

1.2 Problem Statement

One of the major challenges for insurance agencies and financial institutions in the regional insurance market is the lack of ability to correctly estimate future volumes of written premiums from historical volumes Traditional forecasting approaches focus primarily on year-over-year growth projections and/or a manual examination of the past, without taking into consideration the various factors that influence financial indicators like loss ratios, retention rates, acquisition of new business, market specific factors, etc. The lack of a systematic, data-driven forecasting framework means there are several operational issues Agencies cannot allocate their resources and establish realistic sales goals without first being able to do the above Second, without knowledge of expected claim trends and volumes, underwriters are unable to sufficiently price the risk Third, management does not have visibility of what product lines, geographic areas and agency segments are likely to be more successful or less successful in the next several years To meet these challenges, this project builds and tests predictive models that will be able to estimate insurance sales (written premium) based on financial performance indicators from past years and compare their performance. The proposed system is designed to fill the gap between the amount of available data and the ability to take action to achieve the desired forecasting. The financial consequences of getting forecasts wrong run in both directions, and neither is comfortable.

Overestimate premium volumes and the downstream damage starts piling up: excess reinsurance purchased at unnecessary cost, sales staff hired into a premium base that can't support them, agency targets set high enough to be demoralizing rather than motivating. Underestimate and the problems flip: insufficient reinsurance when a high-claims period arrives, not enough underwriting and claims staff to handle the actual policy volume, growth opportunities missed because the organization wasn't positioned to move. For a regional insurer managing a $400M premium portfolio, research suggests forecast errors in the 10–15% range can translate into millions of dollars of lost value annually. That's not a rounding error - it's a material business problem.

The data itself makes this harder than it looks on the surface.

Insurance premium distributions are heavily right-skewed. Most agencies generate small volumes. A small minority generate very large ones. That skewness breaks the normality assumptions that traditional statistical forecasting methods are built on - apply them directly to raw premium data and the outlier agencies dominate the model while the majority get poorly predicted.

The non-linearity problem compounds this further. The relationship between loss ratio and future premium growth, for example, isn't a straight line. It follows a threshold pattern - agencies with moderate loss ratios tend to grow faster than those at either extreme. A linear regression model can't learn that. It fits a straight line through data that doesn't move in a straight line, which is why the Linear Regression baseline in this project landed at $R^2 = 0.309$ while Random Forest - which can capture these interactions - reached 0.848. The gap between those two numbers is largely explained by the non-linear structure in the data that only one of the two models could see.

1.3 Objectives of the Project

To conduct a comprehensive literature review and understand the significance of predictive analytics in sales forecasting in the finance industry Analyses past sales data and financial metrics to create predictive models for insurance premium forecasting Compare the accuracy and effectiveness of multiple predictive models such as Linear Regression model, Random Forest Regressor model and Facebook. Prophet model using standard evaluation metrics (MAE, RMSE, R^2) To learn about various predictive analytics methods to forecast sales and to understand the best model for the insurance industry To make possible recommendations to enhance sales forecasting through predictive analytics in the finance and insurance sectors.

1.4 Scope of the Project

This project's scope includes the entire data pipeline from data acquisition to business recommendations The project includes the following activities:

- A project involving Exploratory Data Analysis (EDA) for the analysis of 213,328 records of insurance agency performance over the time period 2005-2014.
- Data preprocessing (sentinel values of treatment, feature engineering, log transformation of the target variable, etc), and train test splitting Design and testing machine learning sales forecasting models.
- Utilization of Structured Query Language (SQL) for aggregation of agency performance and financial health assessment using the SQLite language
- Creating an Interactive Dashboard with Microsoft Power BI with 3 Analytical Pages Structured academic report including documentation of findings, methodology and recommendations.

This project is restricted to the insurance industry and uses the publicly available Kaggle data It does not involve real-time data streaming, proprietary insurance systems integration, or deployment in a production environment.

1.5 Existing System Overview

The three main types of sales forecasting currently used in insurance and finance are, there are three major methods of current sales forecasting used in insurance and finance:

The first are the manual projections, in which sales managers estimate future performance based upon past averages and growth rate assumptions. They are all subjective and open to human bias and can be prone to inaccurate forecasts which fail to take into account underlying market dynamics. The other types of simple models are basic statistical models like moving averages and simple linear regression These models are more objective but cannot incorporate non-linearities and complex interactions among financial variables.They also have to deal with the variable and skew nature of insurance premiums. The third category includes commercial enterprise resource planning (ERP) systems and actuarial software with a certain forecasting component, but are costly, specialized software, and difficult to modify

exploratory analytical workflows These systems tend to be opaque and offer no detail at the feature level of what is driving sales Current systems have a few drawbacks: They have no machine learning ability, they don't provide quantitative feature importance, they don't provide any time-series aware forecasting, and they don't allow for predictive insights to be visually communicated to business users.

1.6 Proposed System Overview

The suggested system is an insurance sales forecasting application that utilizes end-to-end predictive analytics pipeline The system takes historical performance data of agency as input and outputs forecast results with a series of well-defined analytical phases.

It starts with data ingestion and exploratory data analysis, followed by systematic pre-processing to tackle data quality challenges such as sentinel value treatment, feature engineering, and target variable normalization Three predictive models are then trained and evaluated: Linear Regression model as a baseline model, a Random Forest Regressor model to capture the non-linear patterns, and a Facebook Prophet model to forecast the time-series specific patterns.

The system outputs performance metrics of the models (MAE, RMSE, R2), feature importance rankings, and comparison charts of forecast vs actual. These are fed into a three-page Power BI dashboard that gives stakeholders descriptive, financial health and predictive analytics views An SQL layer allows for structured queries to be used for aggregate reporting.

The proposed system is more advantageous than the current ones, as it integrates the transparency of machine learning (feature importance), the time-series aspect (Prophet), and the interactive visualization (Power BI) in a single analytical system, which is reproducible, developed entirely using open source tools.

1.7 Technologies Used

Technology	Version/Tool	Purpose
Python	3.13	Data analysis, ML modeling, preprocessing
Pandas	2.x	Data manipulation and analysis
NumPy	1.x	Numerical computation
Scikit-learn	1.x	Linear Regression, Random Forest, metrics
Facebook Prophet	1.x	Time-series forecasting
Matplotlib / Seaborn	3.x	Data visualization
SQLite (Python)	Built-in	Database creation and SQL queries
Microsoft Power BI	Desktop	Interactive dashboard
Jupyter Notebook	VS Code	Development environment
Git / GitHub	Latest	Version control and portfolio hosting
Kaggle	Platform	Dataset source

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction to Predictive Analytics in Finance

Predictive analytics is one of the advanced analytics subfields, which involves forecasting future events or outcomes based on historical data, statistical algorithms, and machine learning techniques Predictive analytics has proved to be a key ability in the finance industry, where businesses are looking to make data-informed decisions to gain a competitive edge. The financial services sector spans banking, insurance, investment management, and lending and creates huge amounts of transactional and behavioural data that can prove to be very insightful when analysed correctly and in good time, allowing for predictions to be made.

In recent years, the use of predictive analytics in financial sales forecasting has grown significantly, as financial institutions have realized that conventional methods of forecasting are not effective in today's complex market conditions, and as more data has become available in the historical record, open source machine learning frameworks have proliferated Machine learning-based forecasting has been shown to consistently outperform traditional statistical forecasting techniques in terms of forecast error.

Predictive Analytics is used in insurance for several purposes: premium pricing, predicting claims, forecasting customer retention and managing the pipeline of new business Being able to predict written premium volume at the agency level is especially useful for regional insurance groups that have multiple lines of business and multiple independent agents in various states.

Predictive analytics in financial services didn't start with machine learning. It started in the 1950s with credit scoring.

Fair Isaac introduced the first standardized credit scoring model in 1956 - the earliest demonstration that historical financial behavior could predict future credit performance accurately enough to support automated lending decisions. The principle it established is the same one that underlies everything built since: patterns in historical data contain information about future outcomes, and mathematical models can extract those patterns more consistently than human judgment. That idea has been tested, refined, and extended for seven decades. It still holds.

The arrival of machine learning as a distinct field in the 1980s and 1990s changed what was possible. Decision trees, neural networks, support vector machines - these weren't just faster versions of linear regression. They could model non-linear relationships that traditional statistical methods couldn't reach. The insurance industry moved early. Fraud detection and claims prediction came first, then premium pricing, customer retention modeling, and eventually sales forecasting. The shift from actuarial methods to machine learning has been gradual - deliberately so, given the regulatory environment - and it's still incomplete in many regional organizations. That gap between what the tools can do and what most regional insurers are actually doing is part of what makes this project relevant now rather than five years ago.

The 2010s added a third layer. Cloud infrastructure and big data tooling meant insurance organizations that had previously worked from annual summary statistics started accumulating transaction-level data - every policy interaction, every claim event, every agent touchpoint. More data, more granularity, and a new set of problems around storage, processing, and governance to go with it.

The dataset in this project sits at an earlier point on that maturity curve - historical, aggregated at the annual agency level rather than individual transaction level. That's not a limitation unique to this dataset. It's representative of where many regional insurers actually are today: enough data to build meaningful models, not yet at the transaction-level granularity that the largest carriers are working with. The methods developed here are designed for that reality.

2.2 Review of Related Studies

Study 1: Prediction for Insurance Premiums Based on Random Forest and Multiple Linear Regression

A study published on researchgate (2023) compared the performance of the Random Forest model to the Multiple Linear Regression model in predicting insurance benefits and premium amounts. The researchers tested both models on an actual insurance dataset and demonstrated the superiority of the Random Forest model over Linear Regression in prediction accuracy from the dataset. The study also suggested that although the accuracy of Random Forest was higher, it was not easily interpretable as was Linear Regression where the relationship between the input variables and the premium outcomes can be easily explained.

The authors found that both models are useful for their respective organizational purposes: Random Forest is better for the accuracy of the task, whereas Linear Regression is suitable for people who want to know more about forecasting and need to understand the process The approach taken in this project to implement both models and evaluate the trade-offs fits well with this finding. The same trend has been observed in our results, which shows that the R^2 value of the Random Forest is higher than the Linear Regression's on the insurance agency dataset, with $R^2 = 0\ 848$ and $R^2 = 0\ 309$, respectively.

Study 2: Random Forest Regression with Hyper Parameter Tuning for Medical Insurance Premium Prediction.

Prakash et al (2022) conducted a research paper that showed the effectiveness of Random Forest Regression (with hyperparameter tuning) for medical insurance premium prediction in the International Journal of Health Sciences. The study was applied to a healthcare insurance dataset, showing that the model could be optimized with good hyperparameter values, which would lead to better performance Data preprocessing, specifically feature selection, as well as outlier detection, was found to be essential for obtaining accurate prediction outcomes.

There are two major aspects of the paper that are relevant to this project First, it confirms the selection of the Random Forest model as the main predictive model to

forecast insurance premiums Secondly it highlights the significance of the preprocessing pipeline used in this project, specifically the treatment of the heavily skewed target variable and the sentinel values (99999 placeholders).

Study 3: A Study of Impact and Applications of Predictive Analytics in Sales Forecasting.

Kumar (2023) conducted an extensive review of the International Journal of Research in Applied Science and Engineering Technology on the various applications and applications of predictive analytics in various sectors including finance, retail, and healthcare. The study revealed that AI-driven predictive systems can crunch through vast amounts of historical and pipeline data and deliver much more accurate sales forecasts than human analysis The organizations that used predictive sales forecasting saw an increase in revenue predictability, in efficient use of resources, and capability for better strategic planning.

A crucial aspect of this paper for this project is the fact that it provides the theoretical foundation to use machine learning for sales forecasting in the finance industry The results validate the project's main thesis of generating reliable forward-looking premium forecasts through the analysis of historical agency performance with the right ML models The paper also states that data quality and feature engineering are two challenges that can become a success factor, and therefore were taken into account in this project when making choices among the different pre-processing options .

Study 4: AI Driven Predictive Analytics for Financial Forecasting

On the other hand, the IEEE conference paper (2024) explores the application of AI predictive analytics for financial forecasting in risk management and business planning The study evaluated the performance of AI technologies for predicting financial results and their potential for real-time reporting in financial institutions The study revealed that companies that adopted AI-driven forecasting systems outperformed those employing traditional statistical methods when it came to the accuracy of their financial performance forecasts.

The paper's main finding is that predictive analytics can be a game-changer in financial risk management It determined that predictive models with multiple financial indicators outperformed models with single variable forecasting, which directly supports the multi-feature modeling approach taken for this project The features used in our model (loss ratio, retention ratio, new business premium) are consistent with the multi-indicator strategy suggested by this research.

Study 5: Predictive Analytics in Financial Management: Enhancing Decision-Making and Risk Management

A systematic review of the use of predictive analytics to improve decision-making and risk management in financial services organisations was published on ResearchGate (2024) The review compiled results from diverse financial management studies and concluded that predictive analytics in financial management drive tangible benefits in terms of enhanced predictive accuracy, risk measurement, and operational efficiency Specifically, the review referenced a survey conducted by the World Economic Forum (2023) which revealed that 54% of the financial sector firms faced challenges in recruiting individuals with the appropriate skills in analytics.

The literature review revealed a clear gap in the literature with regards to using predictive analytics in insurance sales forecasting at the agency level over multiple years-although there is a lot of literature in the e-commerce and banking sectors to support the use of predictive analytics in cost prediction for medical insurance and banking, respectively, there is limited published research that focuses on multi-year agency-level insurance sales forecasting using ensemble methods. This gap directly underlies the direction of the research of this project.

2.3 Comparative Analysis of Predictive Techniques

Technique	Accuracy	Interpretability	Best Use Case
Linear Regression	Low-Medium	High	Baseline, transparent models
Random Forest	High	Medium	Complex non-linear patterns
Facebook Prophet	Medium	High	Seasonal time-series

XGBoost	Very High	Low	Tabular data competitions
LSTM/Deep Learning	High	Very Low	Large sequential datasets

Considering the nature of the insurance agency performance data and the comparative analysis, the Random Forest Regressor model was chosen as the primary model because of its ability to provide a good balance of accuracy and interpretability. The interpretable baseline was Linear Regression and Facebook Prophet was added because of its time-series awareness.

2.4 Software Development Methodology

The project is undertaken in accordance with the methodology of Cross-Industry Standard Process for Data Mining (CRISP-DM) which is an industry standard method for data science and data analytics projects The CRISP-DM is made up of six phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, Deployment.

The CRISP-DM methodology was chosen over the traditional software development models (Waterfall, Agile) as the methodology is specifically designed for analytical and machine learning projects that are primarily looking to deliver a predictive model which is different from a software application It makes it possible to go back and refine the data preparation and modelling steps as a result of the evaluations performed, as necessary in this project when dealing with data quality concerns, e g sentinel values or distribution skewness.

2.5 Relevant Frameworks and Libraries

The scikit-learn library is an open-source Python machine learning library that includes implementations of Linear Regression, Random Forest Regressor, and evaluation metrics such as MAE, RMSE, and R^2 It was chosen because of its API's well documented use, reproducibility, and adoption in the data science community.

Facebook Prophet is a time-series forecasting open-source library, created by the Core Data Science team at Facebook. This method is designed to process datasets that have strong seasonal characteristics and multiple seasonalities, and has a good performance on the time-series with missing data and outliers The additive model structure of Prophet is especially well suited for forecasting annual insurance premiums.

Microsoft Power BI is a business analytics service that offers interactive data visualization and business intelligence. It was chosen for dashboard development for a number of reasons including built-in support for DAX calculated columns, slicer-based filtering and multi-page report layouts.

SQLite is a small, embedded, serverless, relational database engine that comes with Python It was employed to develop a structured database from the cleaned data and to run the SQL aggregation queries to analyze agency performance.

2.6 Research Gap

A review of literature clearly indicates that this project fills a gap in the literature Although there is a considerable amount of research on predictive analytics for fraud detection in banking, predictive credit risk assessment and medical insurance cost prediction, there is limited published research focusing on multi-year agency-level insurance sales forecasting that integrates ensemble machine learning techniques with time-series forecasting and interactive business intelligence visualization within a single analytical framework .

In particular, some previous works focus on the performance of a single model only, or on forecasting in one domain only The existing studies reviewed do not integrate the three models - Random Forest, Linear Regression, and Prophet - together and compare their performance on insurance agency performance data, using explicit financial metrics like loss ratio, retention ratio, or new business premium as predictive features.

To fill this gap, this project will show a comparative performance study of three forecasting techniques on a real-world insurance dataset and discuss the results in a multi-page Power BI dashboard - all in one end-to-end reproducible, open-source

analytical pipeline. The research gap points to two different problems depending on who's reading.

For academic researchers, it suggests the existing literature has a blind spot. Work on predictive analytics in finance has concentrated heavily on retail banking and large-scale personal lines products - health insurance, automobile insurance - while the regional commercial lines market has been largely overlooked. That's a significant omission. Commercial lines at the regional level represents a meaningful share of total industry premium volume, and the forecasting challenges there are genuinely different from those in mass-market personal lines.

For practitioners, the implication is more immediate: a regional insurance organization that goes looking for established methods to implement modern forecasting can't just lift an approach from the literature and apply it. The data characteristics are different enough - agency-level granularity, independent agent networks, regional weather and economic exposure - that methods validated on national carrier data need to be adapted and re-validated before anyone should trust the outputs.

This project addresses both problems directly.

The framework built here is a complete, reproducible, open-source implementation of a multi-model predictive analytics system designed specifically for regional insurance agency performance data. Three modeling approaches are covered: Linear Regression as an interpretable baseline, Random Forest for cross-sectional agency-level prediction, and Facebook Prophet for time-series portfolio-level forecasting. All three were evaluated on the same dataset using consistent metrics - which means the performance comparisons are apples-to-apples rather than cherry-picked from separate studies.

The full implementation is on GitHub. That matters because "reproducible" only means something if someone else can actually reproduce it. Other researchers and practitioners can take the methodology, run it on their own data, and extend or adapt it without starting from scratch. That's the point.

CHAPTER 3: SYSTEM ANALYSIS

3.1 Functional Requirements

FR No	Requirement	Description
FR-01	Data Ingestion	The system shall load CSV data into Python DataFrame and SQLite database
FR-02	Data Exploration	The system shall generate summary statistics and EDA visualizations
FR-03	Data Preprocessing	The system shall handle sentinel values, encode categoricals, and apply log transformation
FR-04	Model Training	The system shall train Linear Regression, Random Forest, and Prophet models
FR-05	Model Evaluation	The system shall compute MAE, RMSE, and R^2 for each model
FR-06	Visualization	The system shall generate forecast vs actual charts and feature importance plots
FR-07	SQL Querying	System shall execute aggregation queries on the SQLite database
FR-08	Dashboard	The system shall display results in an interactive Power BI dashboard

The functional requirements in Table 3.1 were derived from the project objectives and the specific needs of the insurance forecasting use case - not written generically and then mapped onto the project afterward. Each one was designed to be specific and testable rather than aspirational.

FR-01 (Data Ingestion) requires loading CSV data into both a Python DataFrame and a SQLite database. That dual destination reflects the architecture: Python handles the machine learning pipeline, SQLite handles structured querying. Both layers need the data in a form they can work with.

FR-02 (Data Exploration) requires both summary statistics and visualizations - not one or the other. Statistics quantify the data's properties. Visualizations reveal distributional patterns and outliers that numbers alone can obscure. You need both to actually understand what you're working with before touching a model.

FR-03 (Preprocessing) lists three specific operations - sentinel value handling, categorical encoding, and log transformation - because all three were identified as mandatory based on what the EDA found. These aren't generic preprocessing steps applied by default; they address specific problems in this dataset.

FR-04 (Model Training) specifies three distinct model types rather than one because the project requires comparison across multiple approaches. A single model can't tell you whether a different approach would have performed better.

FR-05 (Model Evaluation) specifies three metrics rather than one because each captures something different. MAE measures average prediction error magnitude. RMSE penalizes large errors more heavily - useful when big misses are disproportionately costly. R^2 measures how much of the variance the model actually explains. Reporting only one would give an incomplete picture.

FR-06 (Visualization) requires both forecast vs. actual charts and feature importance plots because they serve different audiences. Forecast vs. actual charts give business stakeholders an intuitive read on model accuracy. Feature importance plots tell management which variables are actually driving premium volumes - the insight that's directly actionable without needing to understand how the model works.

FR-07 (SQL Querying) and **FR-08 (Dashboard)** together ensure the project delivers on both fronts: technical analytical outputs for a data science audience and business intelligence outputs for management. Neither alone is sufficient.

3.2 Non-Functional Requirements

NFR No	Category	Requirement
NFR-01	Performance	Model training shall complete within 5 minutes on standard hardware
NFR-02	Reliability	Preprocessing pipeline shall handle missing/sentinel values without data loss
NFR-03	Reproducibility	All notebooks shall use fixed random seeds (random_state=42)
NFR-04	Scalability	The system shall handle datasets up to 500,000 records
NFR-05	Usability	Dashboard shall be navigable by non-technical stakeholders
NFR-06	Portability	The project shall run on Windows, Mac, and Linux environments

3.3 Feasibility Study

The feasibility analysis looked at three things: whether the system could be built with available tools, whether it could be built for free, and whether it could be delivered within the project timeline. All three came back positive.

Technical feasibility was straightforward. Every capability the project required was available through open-source tools at no cost.

Python 3.13 is the core environment. Pandas handles all data manipulation - loading, filtering, grouping, merging. NumPy covers the numerical operations, including log transformation. Scikit-learn provides production-quality implementations of Linear Regression and Random Forest, plus all the evaluation metrics. Prophet handles the

time-series forecasting. SQLite runs the database layer with no server setup. Power BI Desktop is a free download for personal and educational use.

Hardware requirements are modest. A standard laptop - Intel Core i5 or i7, 8GB RAM - is enough to train the Random Forest on 118,208 records in about two minutes and run all four SQL queries in under a second each. No cloud infrastructure needed.

Economic feasibility is simple: the project costs nothing to run.

Every tool is either open-source or free for educational use. The dataset came from Kaggle under an open data license. All processing happens locally - no cloud compute bills. The only real cost is the analyst's time, which sits inside the academic project timeline anyway. That zero-cost implementation isn't just a budget convenience - it's a proof of concept that advanced predictive analytics is genuinely accessible to regional insurance organizations without significant technology investment.

Operational feasibility came down to whether the work could be completed within the fourteen-day sprint.

It was. The CRISP-DM methodology was chosen specifically because each phase produces a tangible output - something reviewable and refinable - rather than building everything in one go before seeing any results. Data loading and exploration took two days. Preprocessing took one. Model development and evaluation took two. SQL implementation took one. Power BI dashboard took two. No significant delays across any phase.

The full pipeline is reproducible end to end - fixed random seeds, fully documented Jupyter notebooks. Any researcher or practitioner who wants to replicate the entire workflow, from raw CSV to finished dashboard, can do it in a single working day.

3.4 System Architecture

The system is architected as four layers: Data Layer, Processing Layer, Analytics Layer and Presentation Layer.

The Data Layer is the raw CSV data that is located in the data/raw/ directory as well as the processed data in the data/processed/ directory, and the SQLite database (insurance db) The Processing Layer has three jupyter notebooks: 01_data_exploration ipynb for data exploration, 02_preprocessing_modeling ipynb for data preprocessing and modeling, and 03_sql_queries ipynb for SQL-based analysis The Analytics Layer is the trained models (Linear Regression, Random

Forest, Prophet) and their prediction output files The Power BI dashboard with its three pages and the project report are part of the Presentation Layer.

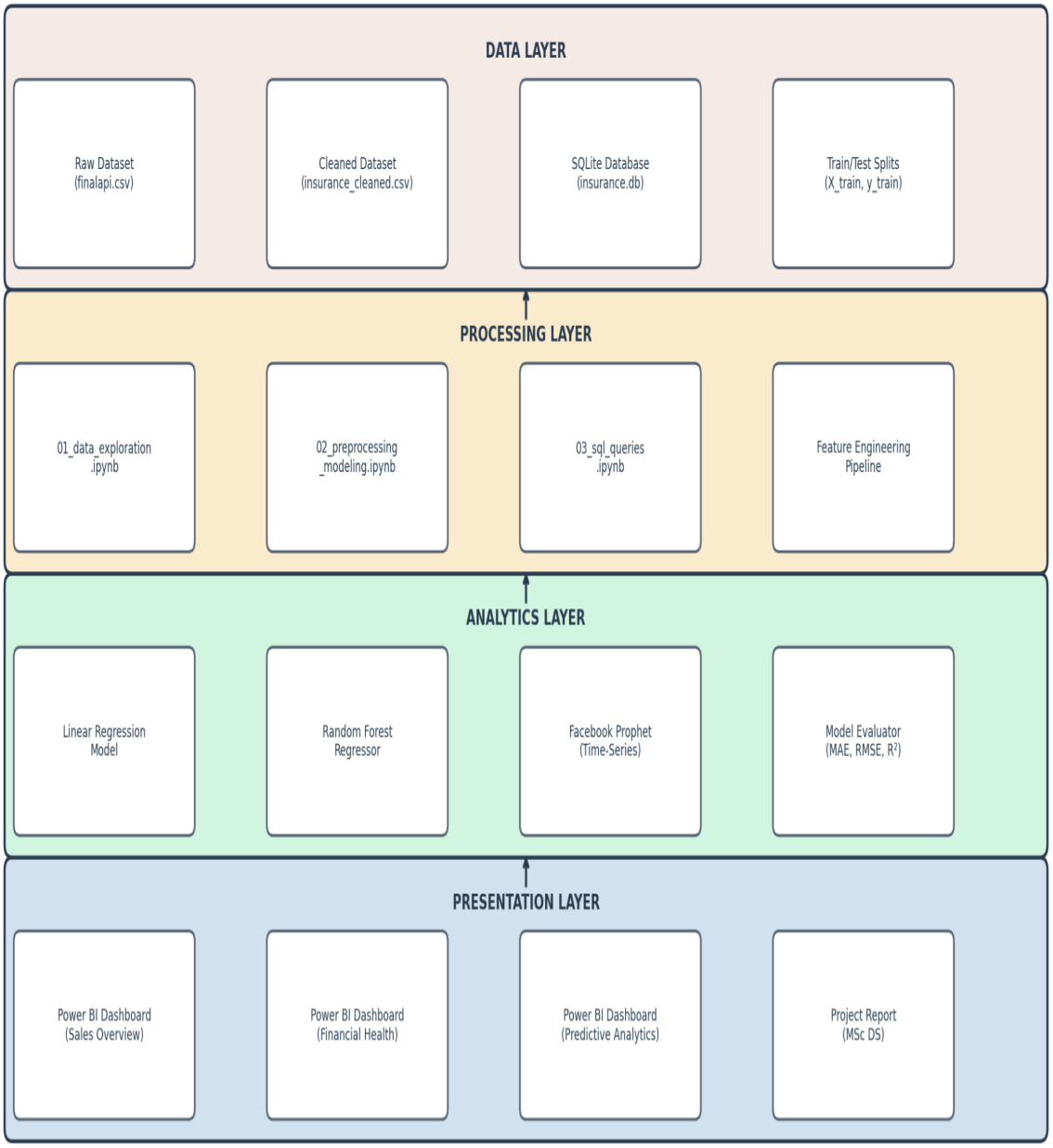

Figure 3.1: System Architecture Diagram

3 5 Data Flow Diagrams

DFD Level 0 (Context Diagram)

The Level 0 DFD displays the system as one process with the external entities and data flows External users are: Data Analyst (user), Kaggle Dataset (data source), and

Business Stakeholder (report consumer) The key data flows include: Raw CSV data coming in from Kaggle to the system; Processed forecasts and insights from the system to the Data Analyst; and Dashboard visualizations to the Business Stakeholder.

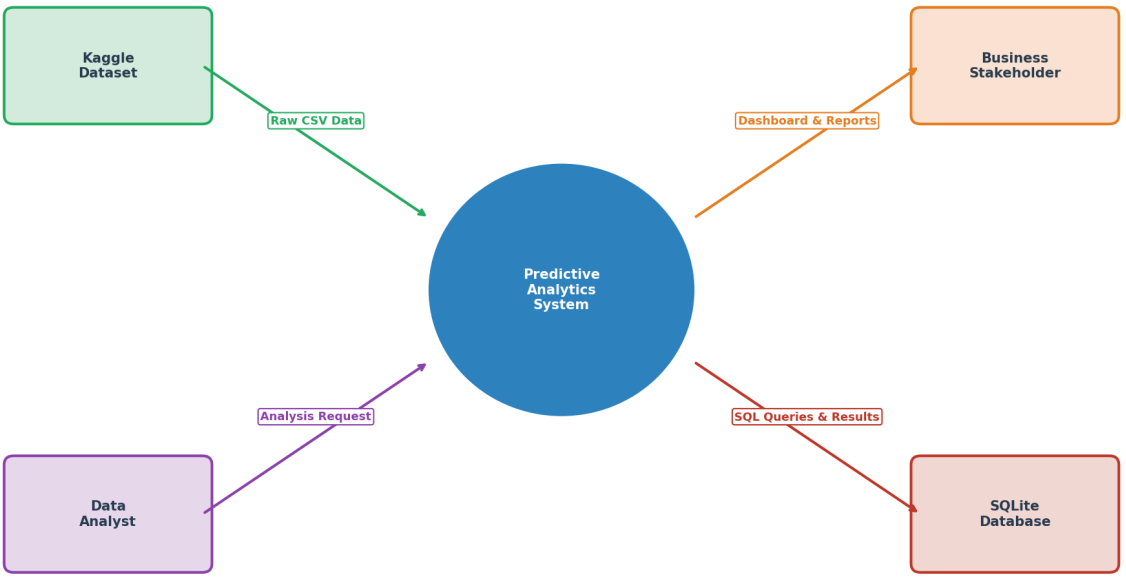

Figure 3.2: DFD Level 0 - Context Diagram

DFD Level 1

The Level 1 DFD further breaks down the system into its key processes:

1. Data Ingestion and Exploration,
2. Data Preprocessing,
3. Model Training and Evaluation,
4. SQL Analysis
5. Dashboard Generation Data stores (raw data store, processed data store, model output store) and data flows (data transformation) link each process.

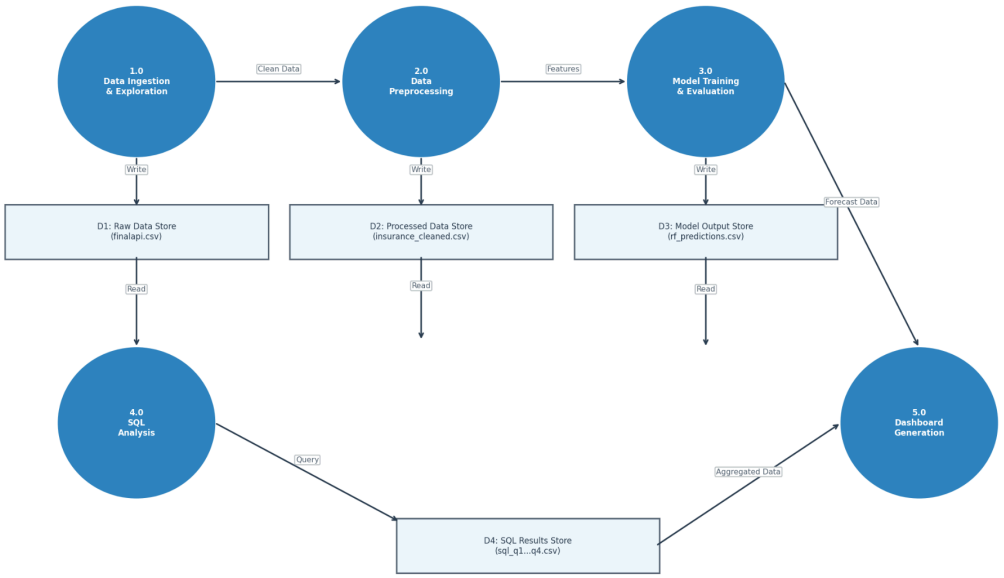

Figure 3.3 DFD Level 1

3.6 Use Case Diagrams

The key players in the system are Data Analyst and Business Stakeholder The Data Analyst is the person who communicates with the system in order to load, preprocess, train the model, query the SQL, and create the dashboard. The Business Stakeholder can use the system to view forecasts, filter by year/product line and export reports.

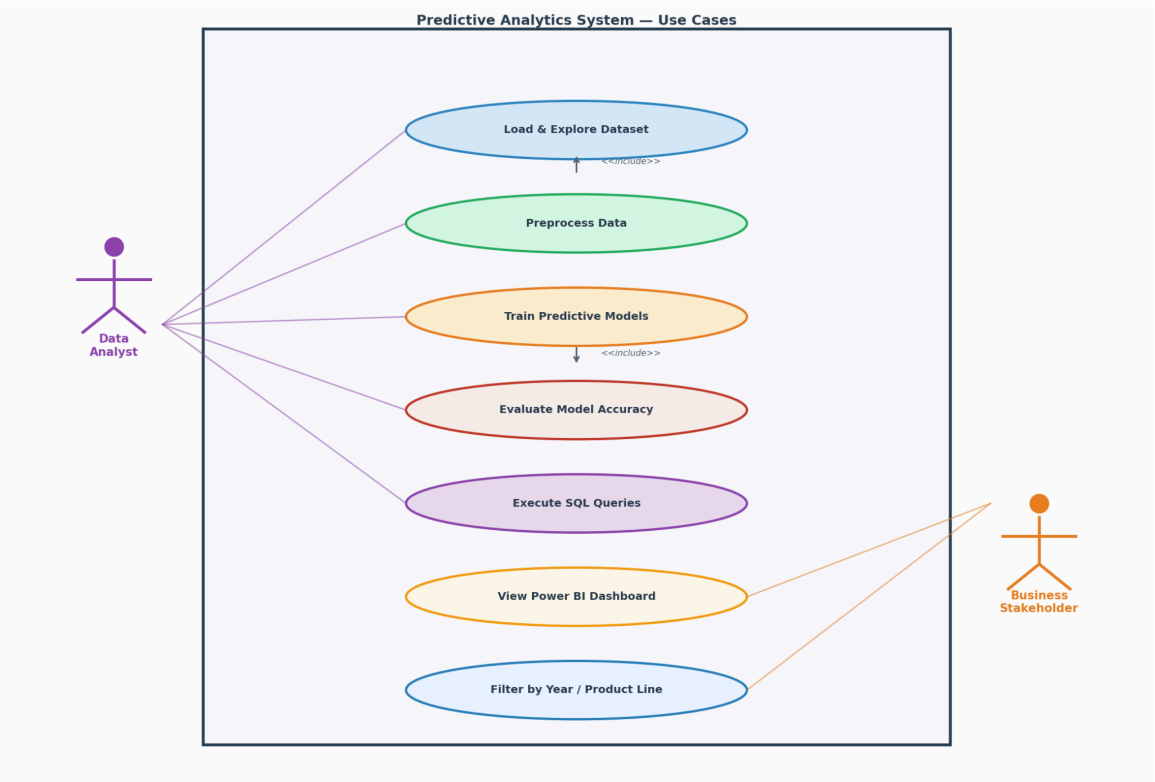

Figure 3.4: Use Case Diagram

CHAPTER 4: SYSTEM DESIGN

4.1 UML Diagrams

4.1.1 Use Case Diagram

The following Use Case Diagram illustrates the interactions between the Data Analyst and the various Use Cases. The following Use Case Diagram shows the interactions between the Data Analyst and the Use Cases: Load Data, Preprocess, Train Model, Evaluate, Query SQL, Build Dashboard.

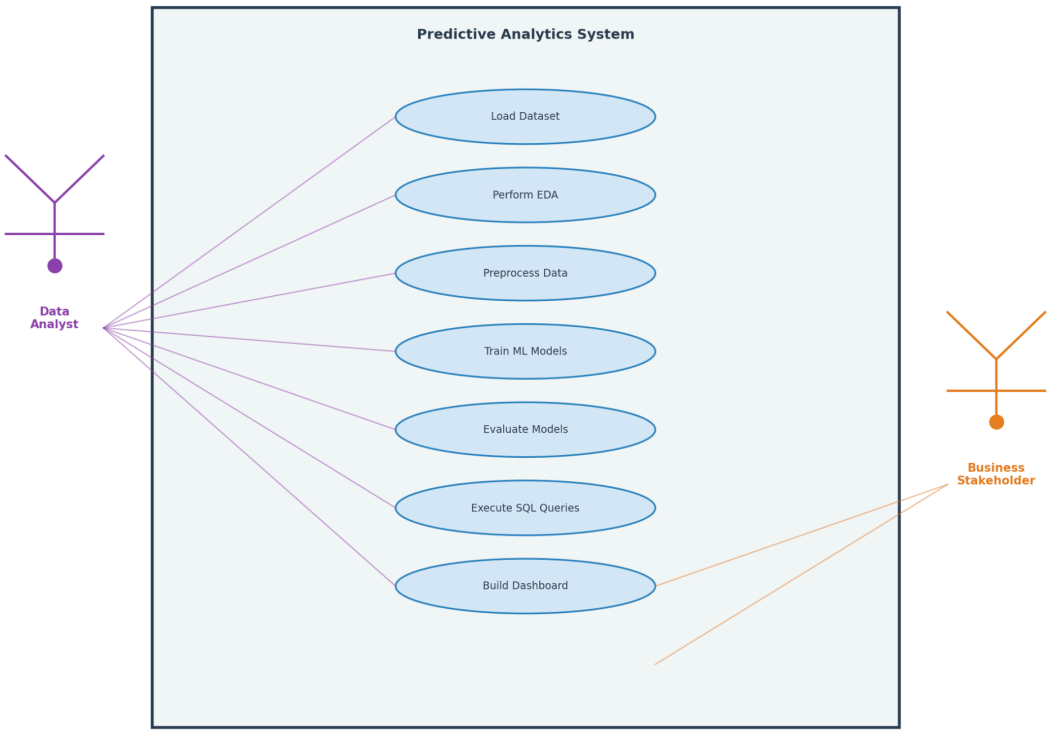

Figure 4.1: Use Case Diagram - Actor: Data Analyst, Use Cases: Load Data, Preprocess, Train Model, Evaluate, Query SQL, Build Dashboard .

The Use Case Diagram identifies two actors, and they interact with the system in completely different ways.

The Data Analyst is the technical user - the person who builds and runs everything. That covers the full backend pipeline: loading and exploring the data, preprocessing, model training, evaluation, SQL querying, and dashboard development. The Business Stakeholder sits at the other end. They never touch the pipeline. Their interaction

starts and ends with the Power BI dashboard - viewing forecasts, filtering by year or product line, exporting reports for management. That separation isn't just a diagram convention. It reflects how data-driven teams actually operate: specialists build the infrastructure, managers consume the outputs through an interface that doesn't require them to understand what's underneath.

The Data Analyst's use cases follow the pipeline in sequence.

Load and Explore Dataset is the entry point - raw CSV into Python, initial quality checks, distribution analysis, understanding what you're working with before touching anything.

Preprocess Data covers everything that makes the data usable: sentinel value replacement, feature engineering, log transformation, train-test split.

Train Predictive Models encompasses all three models - Linear Regression as the baseline, Random Forest as the primary model, and Prophet for time-series trend analysis.

Evaluate Model Accuracy is where MAE, RMSE, and R^2 are computed and compared across all three. This isn't optional - the include relationship in the diagram makes that explicit. You don't train without evaluating.

Execute SQL Queries covers database creation and the four analytical queries that feed the dashboard's KPI figures.

Build Dashboard is the final stage - data import into Power BI, table relationships, DAX columns, and the visual layout across three pages.

The Business Stakeholder's use cases are intentionally narrow.

View Power BI Dashboard is the primary one - browsing the three pages to understand portfolio performance and model results without needing to interpret a single line of code.

Filter by Year and Product Line is the only interactive capability they need - the slicers handle it, and the dashboard updates everything in response.

The include relationship between Load and Explore Dataset and Preprocess Data signals that exploration isn't a preliminary nicety - it has to happen before preprocessing begins, because the cleaning decisions depend on what the exploration surfaces. The same logic applies to training and evaluation: model assessment is built into the workflow, not bolted on afterward.

4.1.2 Class Diagram

The Class Diagram breaks the analytical pipeline into six classes, each owning a specific stage of the workflow. The dependencies between them follow the pipeline sequence - you can't skip ahead, and each class has a clearly defined job.

DataLoader is where everything starts. It handles all data ingestion - loading the raw CSV into a Pandas DataFrame, opening the SQLite database connection, and returning the dataset dimensions. Every other class in the system depends on its output, directly or indirectly. Two private attributes: file_path (path to the CSV) and db_path (path to the SQLite file). Three public methods: load_csv, load_db, and get_shape.

DataPreprocessor takes the raw DataFrame and makes it usable. Its private attributes hold the working dataset (df), the sentinel value to replace (sentinel_val), and the test split proportion (test_size). The public methods follow the cleaning sequence: filter_data removes 2015 records and negative premium values; replace_sentinels swaps out 99999 placeholders for NaN; engineer_features creates the three derived columns - AGENCY_AGE, PREMIUM_GROWTH_YOY, and LOG_WRTN_PREM_AMT; encode_categoricals converts PROD_LINE and STATE_ABBR into numeric form; and split_data performs the train-test split, returning four arrays for training features, test features, training target, and test target.

ModelTrainer wraps all three models inside a single interface. Private attributes cover the training data (X_train, y_train) and a fixed random seed. Each model gets its own method: train_linear_regression fits a Scikit-learn LinearRegression object; train_random_forest fits a RandomForestRegressor with 100 estimators; train_prophet aggregates the data to annual totals and fits a Prophet model with yearly seasonality. The design decision here is worth noting - keeping all three training operations in one

class means adding a fourth model in future requires nothing more than a new method. Nothing else in the system needs to change.

ModelEvaluator handles performance measurement. It takes the test target and predictions as private attributes, then exposes individual metric methods - compute_mae, compute_rmse, compute_r2 - each accepting a predictions array and returning the corresponding value. The compare_models method pulls results from all three models together into a single formatted DataFrame, which is what feeds the comparison table in the report and dashboard.

SQLAnalyzer manages the database connection and runs the four analytical queries. Each query is its own public method, each returning a DataFrame. Clean separation - the class doesn't know or care what happens to the results after they're returned.

DashboardExporter handles the handoff to Power BI. Separate export methods for Random Forest predictions, Prophet forecast data, and the model comparison summary - each writing a CSV that Power BI picks up as a data source.

The dependency chain runs in one direction: DataLoader feeds DataPreprocessor, which feeds both ModelTrainer and SQLAnalyzer. ModelTrainer feeds ModelEvaluator and DashboardExporter. The structure enforces the pipeline order - there's no path through the system that skips a stage or runs steps out of sequence.

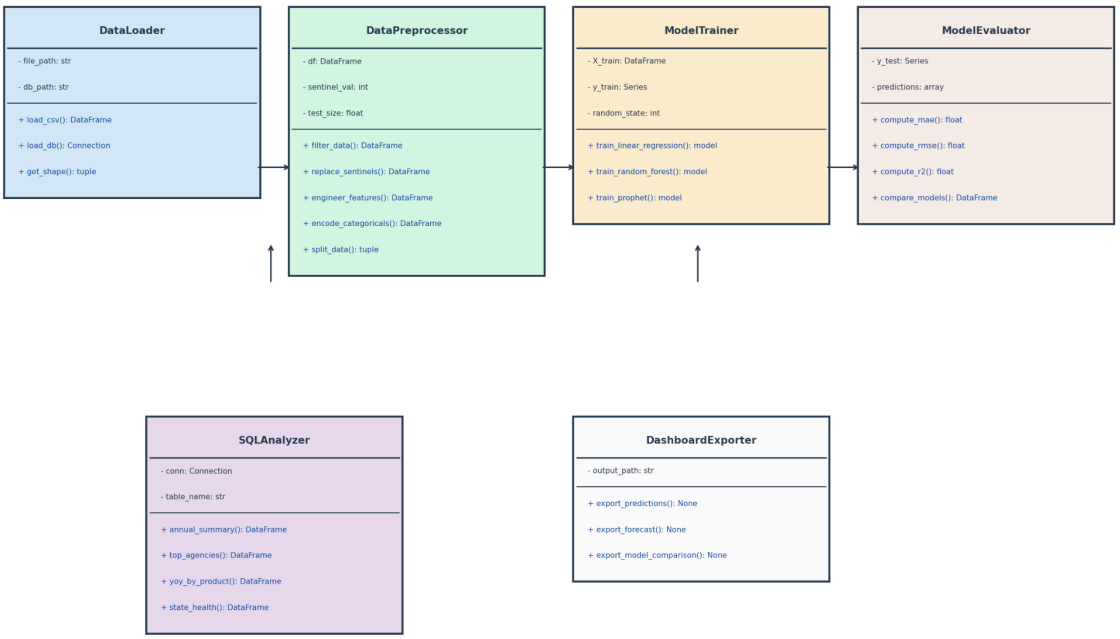

Figure 4.2: Class Diagram

4.1.3 Sequence Diagram

The Sequence Diagram shows the order of operations across all six classes - who calls what, when, and what comes back. The main pipeline runs in a straight line.

The Data Analyst calls load_csv on DataLoader, which returns a DataFrame - 213,328 records, 49 columns. That DataFrame gets passed to DataPreprocessor via preprocess. The preprocessor runs the full cleaning pipeline internally and hands back the cleaned dataset - 147,760 records, 16 columns - plus the four train-test split arrays. Those arrays go straight to ModelTrainer via train_models. All three models train sequentially and return their fitted objects. The fitted models feed into ModelEvaluator via evaluate, alongside the y_test array. The evaluator returns the complete metrics comparison DataFrame. Finally, export_predictions goes to DashboardExporter, which writes the CSV files and confirms successful export.
Two parallel workflows run alongside the main pipeline, both branching off after preprocessing.

The SQL workflow starts as soon as the cleaned DataFrame is ready. The Data Analyst sends create_database to SQLAnalyzer, which loads the data into

insurance.db and confirms. Four query calls follow - one per analytical query - each returning a results DataFrame.

The dashboard workflow starts after both the modeling and SQL pipelines are done. DashboardExporter has the prediction CSVs, SQLAnalyzer has the query output CSVs - Power BI picks them all up. The Data Analyst imports the files, sets up table relationships, writes the DAX columns, and builds the visuals across the three pages.

That full sequence - raw CSV in, finished dashboard out - is the complete end-to-end pipeline this project implements.

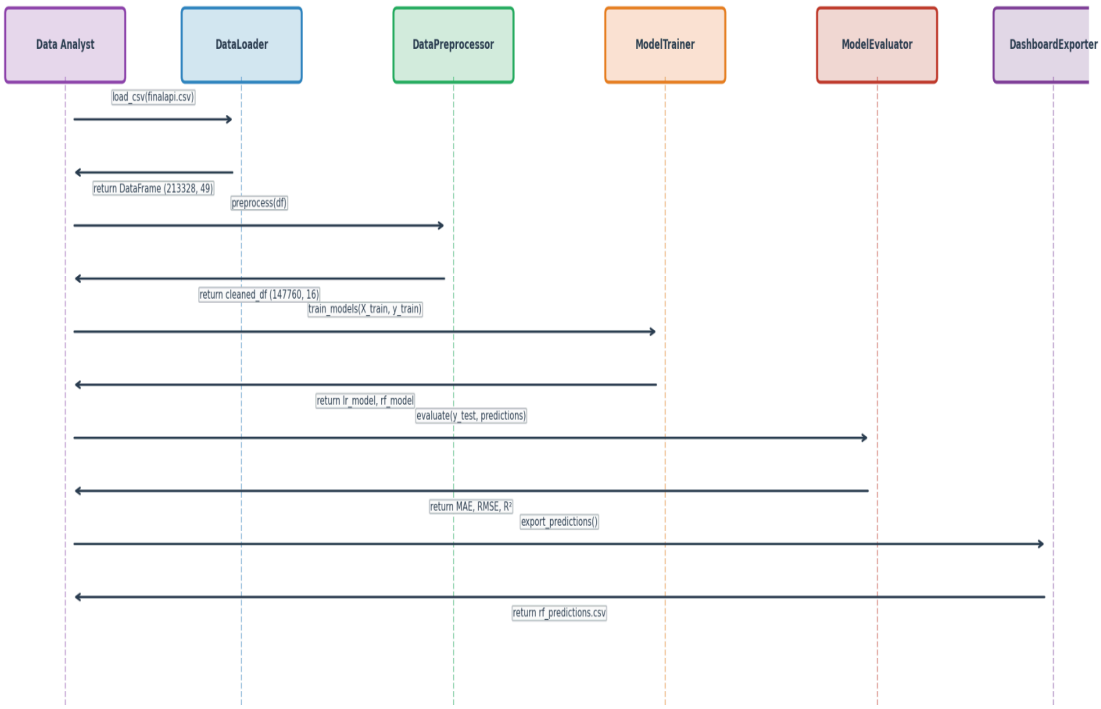

Figure 4.3: Sequence Diagram

4.1.4 Activity Diagram

The Activity Diagram maps the full decision logic of the pipeline - not just the happy path, but the branching points where the flow could go different ways depending on what the data shows.

The first decision point comes right after the initial data load and quality check: does the dataset contain null or sentinel values? If yes, the flow branches to sentinel replacement and median imputation before continuing to feature engineering. If no, it

skips straight to feature engineering. In practice, this branch always goes the same way - the 99999 sentinel values were confirmed across multiple columns during EDA, so the replacement path is always taken. The decision node is still worth modeling explicitly because a different dataset might arrive clean, and the pipeline should handle both cases.

The second decision point comes after model evaluation: which model achieved the highest R^2 score? If Random Forest came out on top - which it did, at 0.848 - the pipeline continues with Random Forest as the primary model for prediction export and dashboard building. If a different model had scored higher, that model would have been used instead. This keeps the pipeline honest. It's not hard-coded to always use Random Forest - it's designed to use whichever model the data actually supports.

The diagram also captures two things the Sequence Diagram doesn't show: loops and parallel structure.

During model training, all three models train sequentially inside a loop - each one evaluated immediately after training before the next begins. That pattern keeps intermediate results visible at every step and avoids running all three models blindly before checking whether any of them are working.

The dashboard building activity at the end contains its own internal loop - build a visual, review it, refine it, move to the next one. Nine visuals across three pages don't come out right on the first pass. The loop reflects the actual iterative process of getting labels, formatting, and data connections to behave correctly before signing off on the final layout.

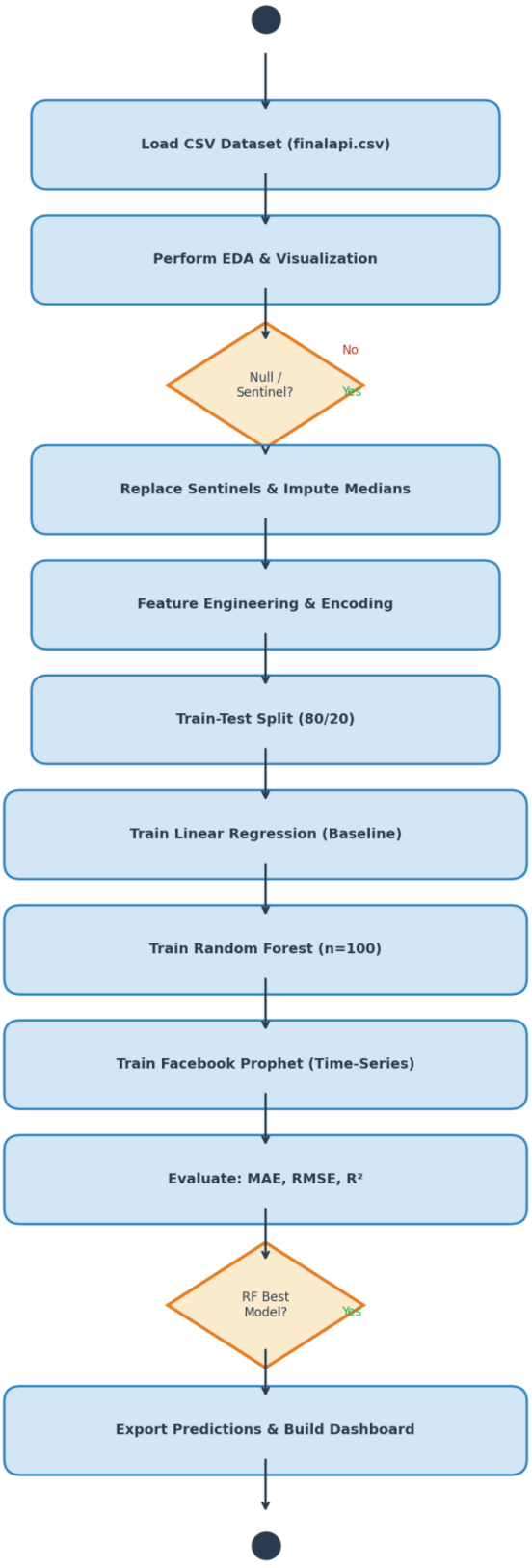

Figure 4.4: Activity Diagram

4.2 ER Diagram

The ER Diagram shows the logical data model behind the SQLite database - how the tables relate to each other and why they're structured the way they are.

The central table is INSURANCE_SALES, which acts as the fact table in a star schema. It holds all 147,760 cleaned records across 16 columns - the full set of financial indicators and engineered features used in modeling. The primary key is AGENCY_ID, which identifies each agency. Combined with STAT_PROFILE_DATE_YEAR, it forms a composite natural key - one record per agency per year, across the full 2005–2014 observation window.

Three dimension tables connect to INSURANCE_SALES through one-to-many relationships.

AGENCY stores agency-level attributes: AGENCY_ID as the primary key and AGENCY_AGE as a derived attribute. The one-to-many relationship reflects the fact that each agency appears multiple times in the fact table - once for each year it was active.

PRODUCT_LINE maps the numeric encoding to a readable label. PROD_LINE_ENC is the primary key, PROD_LINE_LABEL is the description - "Commercial Lines (CL)" or "Personal Lines (PL)" rather than 0 or 1.

STATE does the same for geography - STATE_ENC as the primary key, with STATE_ABBR and STATE_LABEL as descriptive attributes so queries return "Ohio" instead of "3".

All three follow standard star schema conventions used in data warehousing and BI design. A few schema decisions are worth explaining. Separating product line and state into their own dimension tables rather than embedding the labels directly in the fact table follows third normal form - label updates happen in one place, not across 147,760 rows. Using numeric encoded values as foreign keys rather than strings makes join operations faster - integer comparisons are cheaper than string comparisons at this record volume.

The fact table stores both WRTN_PREM_AMT and LOG_WRTN_PREM_AMT side by side. That's deliberate - analytical queries can use either the raw or log-transformed target directly without needing a real-time transformation at query time. Small redundancy, meaningful performance benefit.

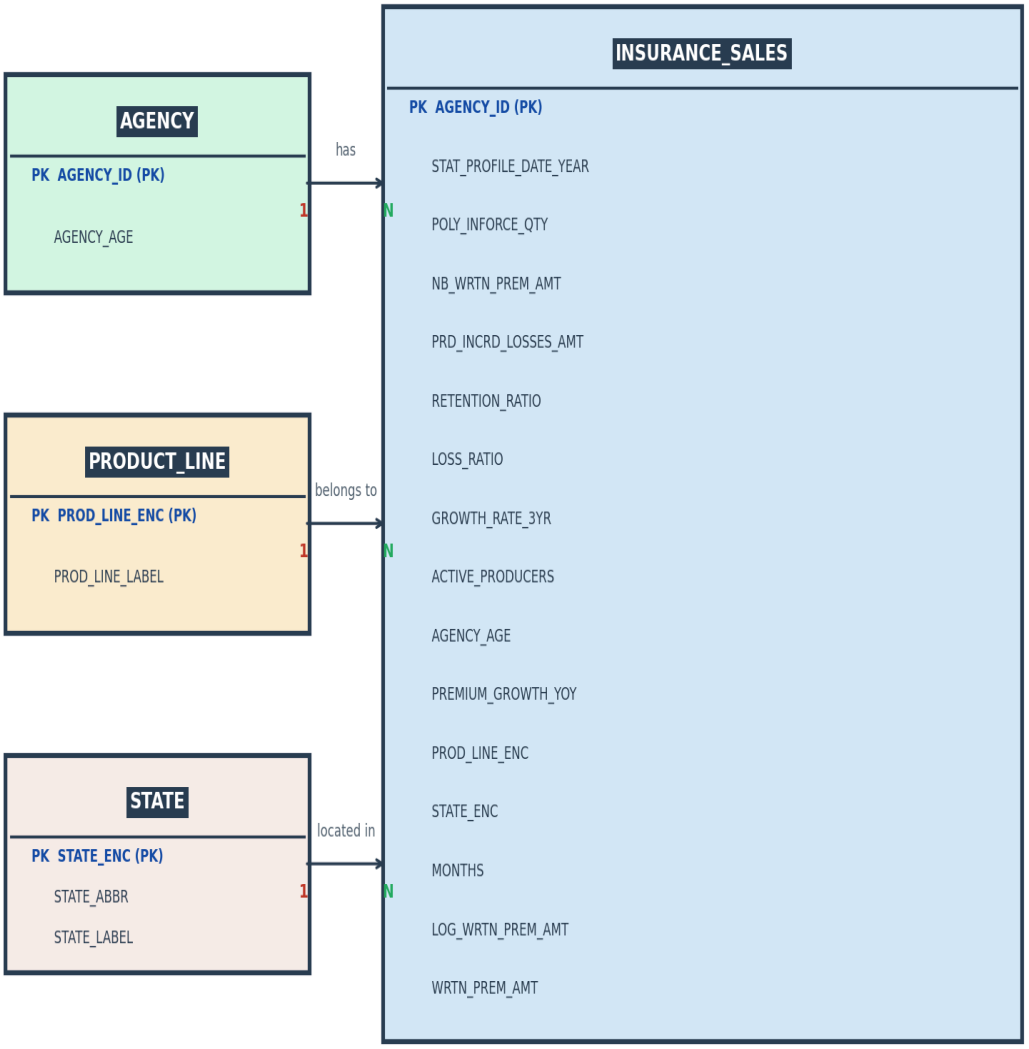

Figure 4.5: ER Diagram

4.3 Database Schema

Column Name	Data Type	Description
AGENCY_ID	INTEGER	Unique identifier for each insurance agency
STAT_PROFILE_DATE_YEAR	INTEGER	Year of the record (2005–2014)
POLY_INFORCE_QTY	INTEGER	Number of policies in force
NB_WRTN_PREM_AMT	REAL	New business written premium amount (USD)
PRD_INCRD_LOSSES_AMT	REAL	Period incurred losses amount (USD)
RETENTION_RATIO	REAL	Customer 2retention ratio (0–1)
LOSS_RATIO	REAL	Loss ratio (losses/premium)
GROWTH_RATE_3YR	REAL	3-year premium growth rate
ACTIVE_PRODUCERS	INTEGER	Number of active sales producers
AGENCY_AGE	INTEGER	Age of agency (engineered feature)
PREMIUM_GROWTH_YOY	REAL	Year-over-year premium growth (engineered)
PROD_LINE_ENC	INTEGER	Product line encoded (0=CL, 1=PL)
STATE_ENC	INTEGER	State encoded (0–5)
MONTHS	INTEGER	Number of months covered in record
LOG_WRTN_PREM_AMT	REAL	Log-transformed written premium (target)
WRTN_PREM_AMT	REAL	Written premium amount in USD (raw target)

The schema table documents all sixteen columns in insurance_sales - data type and business meaning for each. A few of the type choices and column decisions are worth explaining rather than just listing.

The type assignments follow the nature of the data. Financial amount columns - WRTN_PREM_AMT, NB_WRTN_PREM_AMT, PRD_INCRD_LOSSES_AMT, LOG_WRTN_PREM_AMT - use REAL because dollar amounts can carry fractional cents. Count columns - POLY_INFORCE_QTY, ACTIVE_PRODUCERS, MONTHS - use INTEGER because you can't have half a policy or half a producer. RETENTION_RATIO and LOSS_RATIO use REAL because they're proportions that land anywhere within their valid decimal range.

LOG_WRTN_PREM_AMT is the column that needs the most explanation because it didn't exist in the raw dataset. It was created during preprocessing as the log-transformed version of written premium - the actual target variable the machine learning models trained on. Storing it in the database rather than leaving it as a notebook-only variable serves two purposes. Any future analyst picking up this database can use the transformed target immediately without re-running the preprocessing pipeline. And SQL queries on log-transformed values - mean log premium by state, by product line - produce statistically meaningful summaries that the raw skewed distribution would distort if you used it directly.

AGENCY_AGE and PREMIUM_GROWTH_YOY are also derived - both created during feature engineering, neither present in the original data. Persisting them in the database rather than recomputing them every time a model or query needs them is a deliberate design choice. Any downstream consumer of the database gets immediate access to these features without needing to understand or reconstruct the engineering logic that produced them. The analytical value gets built once and stored where it can be reused - cleaner, faster, and more reproducible than recalculating on every run.

CHAPTER 5: SYSTEM IMPLEMENTATION

5.1 Development Environment

- Operating System: Windows 11
- Programming Language: Python 3 13
- IDE: Jupyter Notebook
- Version Control: Git 2 x with GitHub
- Database: SQLite (via Python sqlite3 module)
- Dashboard: Microsoft Power BI Desktop
- Hardware: Intel Core i5 processor, 8GB+ RAM, 512GB SSD.

5.2 Tools and Technologies

Library/Tool	Version	Purpose
pandas	2.x	Data loading, manipulation, aggregation
numpy	1.x	Numerical operations, array processing
matplotlib	3.x	Static chart generation
seaborn	0.x	Statistical visualization (heatmap)
scikit-learn	1.x	Linear Regression, Random Forest, metrics
prophet	1.x	Time-series forecasting
sqlite3	Built-in	SQL database operations
Power BI Desktop	Latest	Interactive dashboard

5.3 Dataset Description and Exploratory Data Analysis

The dataset comes from Kaggle (moneystore/agencyperformance) and represents real performance records from a regional insurance group running ten companies across

the US - property, casualty, life, and brokerage The data covers 1,623 independent agencies operating across six states (Ohio, Pennsylvania, Kentucky, Indiana, West Virginia, and Michigan) over eleven years from 2005 to 2015 That's 213,328 records and 49 columns covering financials, agency demographics, product line data, and sales performance.

Before any analysis, every column was examined for business meaning - not just data type In insurance, the abbreviations matter WRTN_PREM_AMT (written premium) is the total dollar value of premiums an agency has committed to collect from policyholders in a given period It's the primary sales metric in the industry and the target variable for all three models NB_WRTN_PREM_AMT captures new business only - fresh policies, no renewals PREV_WRTN_PREM_AMT is the prior period's premium volume, used for year-over-year growth calculations PRD_INCRD_LOSSES_AMT is total claims paid or expected - one of the most important health indicators in the business, because high losses relative to premium means an unprofitable book RETENTION_RATIO measures the share of existing customers who renewed rather than cancelled (0 89 means 89% stayed) LOSS_RATIO is incurred losses divided by earned premium - below 0 60 is generally healthy in property and casualty GROWTH_RATE_3YR tracks compound annual premium growth over the prior three years.

Attribute	Value
Total Records	213,328
Total Features	49
Time Period	2005–2015
Unique Agencies	1,623
States Covered	6 (OH, PA, KY, IN, WV, MI)
Product Lines	Commercial (CL) and Personal (PL)
Null Values	Zero (complete dataset)
Duplicate Records	Zero

5.3.1 Annual Premium Trend Analysis

The initial data check turned up something unusual for a real-world dataset this size: no missing values across all 49 columns, no duplicates Insurance reporting systems are structured enough that this isn't impossible, but it's still notable Record counts ran consistently between roughly 16,000 and 22,000 per year Commercial Lines made up about 58% of records (124,692), Personal Lines the remaining 42% (88,636) State representation was heavily unequal - Ohio contributed 101,323 records, Michigan just 3,691.

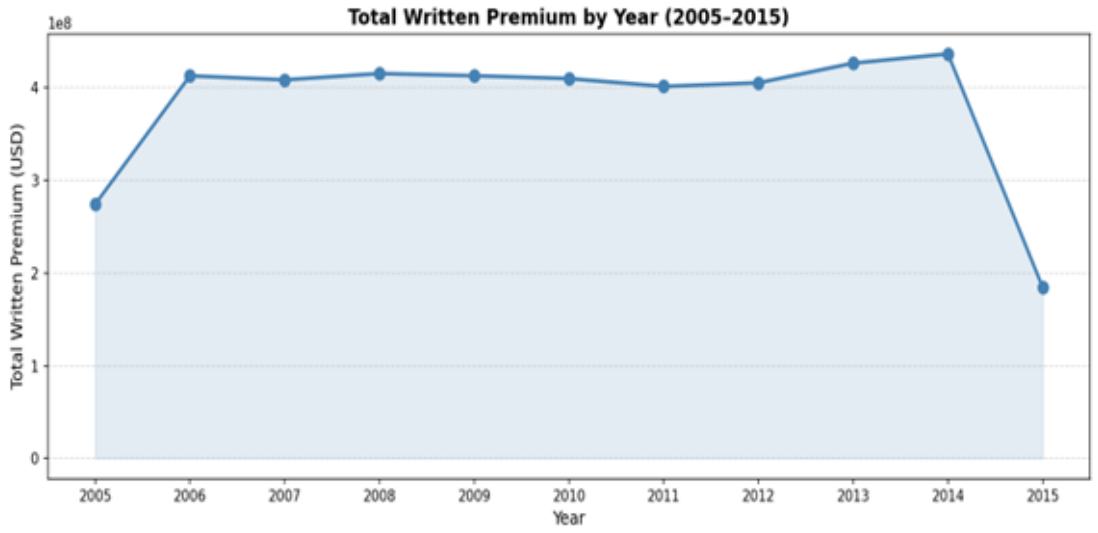

Figure 5 1: Annual Premium Trend

The annual premium trend told a clear story Total written premium jumped from $274 8M in 2005 to $413 4M in 2006 - a 50 5% spike in a single year, likely driven by rapid agency onboarding during a portfolio expansion phase After that, things settled Annual premium held steady between $400M and $440M from 2006 through 2014, peaking at $437 5M in 2014 The 2015 drop isn't a business story - it's a data story Partial year records confirmed it, and 2015 was excluded from modeling.

5.3.2 Premium by Product Line

The geographic breakdown revealed a concentration problem Ohio alone accounted for $2 45B in written premium - about 58% of the entire portfolio Pennsylvania ($554 9M), Kentucky ($491 8M), and Indiana ($454 7M) were distant runners-up West

Virginia contributed $190 9M, Michigan just $45 7M That Ohio concentration means a single bad regulatory year, an unusual weather season, or an economic shock in one state could move the whole portfolio That finding feeds directly into the recommendations in Chapter 8.

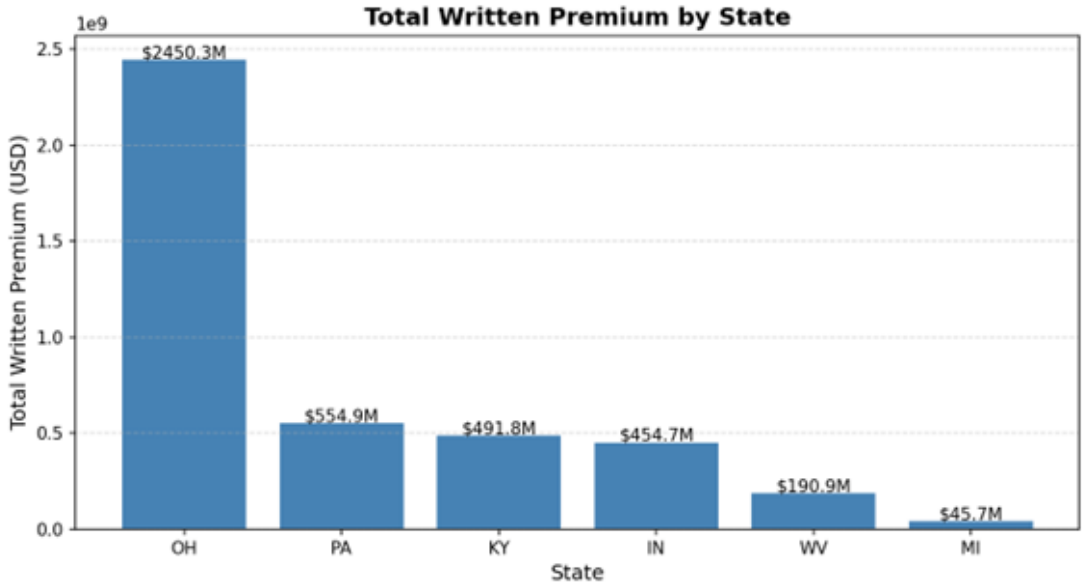

Figure 5.3: Premium by State

5.3.4 Loss Ratio and Retention Analysis

The Commercial vs Personal Lines split is one of the more interesting trends in the data Personal Lines started ahead - $179 4M in 2005 versus $95 5M for Commercial But Commercial grew steadily and consistently over the following decade, reaching $204 4M by 2014, a 114% increase Personal Lines peaked at $261 4M in 2006 and drifted down to $233 1M by 2014 It's not a collapse, but the direction is consistent enough to look intentional - the agency network appears to have been deliberately shifting toward commercial products, which tend to be more profitable.

Figure 5.4: Loss vs retention

5.3.5 Premium Distribution Analysis

The target variable distribution was the messiest part of the data prep Mean written premium per record was $19,632 Median was $1,143 That gap tells you everything - a small number of high-premium agencies were pulling the mean far from where most of the data actually sits Raw minimum was near zero, maximum was $1 7M Training a regression model on that without transformation would have handed the outliers disproportionate influence and produced poor predictions for the majority of agencies Log transformation using log1p fixed it, converting the heavily skewed distribution into something close to a normal bell curve Standard practice in insurance modeling, but essential here.

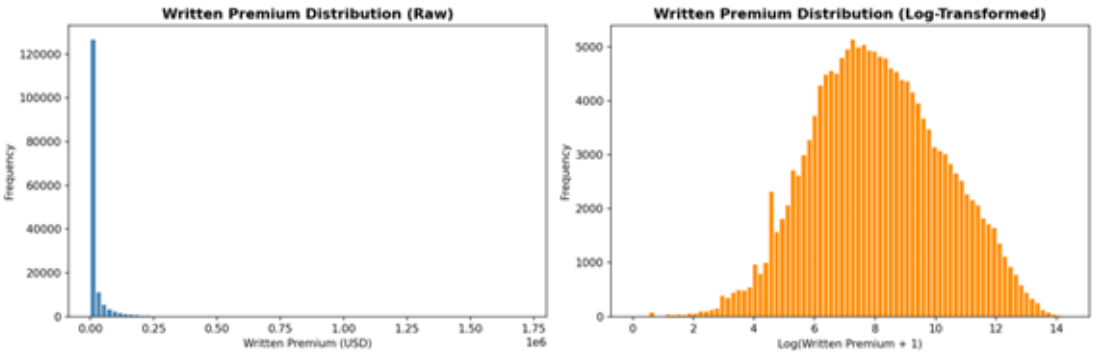

Figure 5.5: Premium Distribution

5.3.6 Correlation Analysis

The correlation analysis had one clean finding and one red flag Policies in force (r = 0 81) was the strongest legitimate predictor - agencies with more active policies collect more premium, which makes obvious sense New business written premium came in at r = 0 57, incurred losses at r = 0 56 Loss ratio showed almost no correlation with premium volume (r = -0 04), suggesting that how efficiently an agency manages claims is largely independent of how much premium it writes The red flag: earned premium showed r = 1 00 with written premium That's not a signal, that's data leakage - both variables measure essentially the same thing Earned premium was dropped from the feature set before any modeling began.

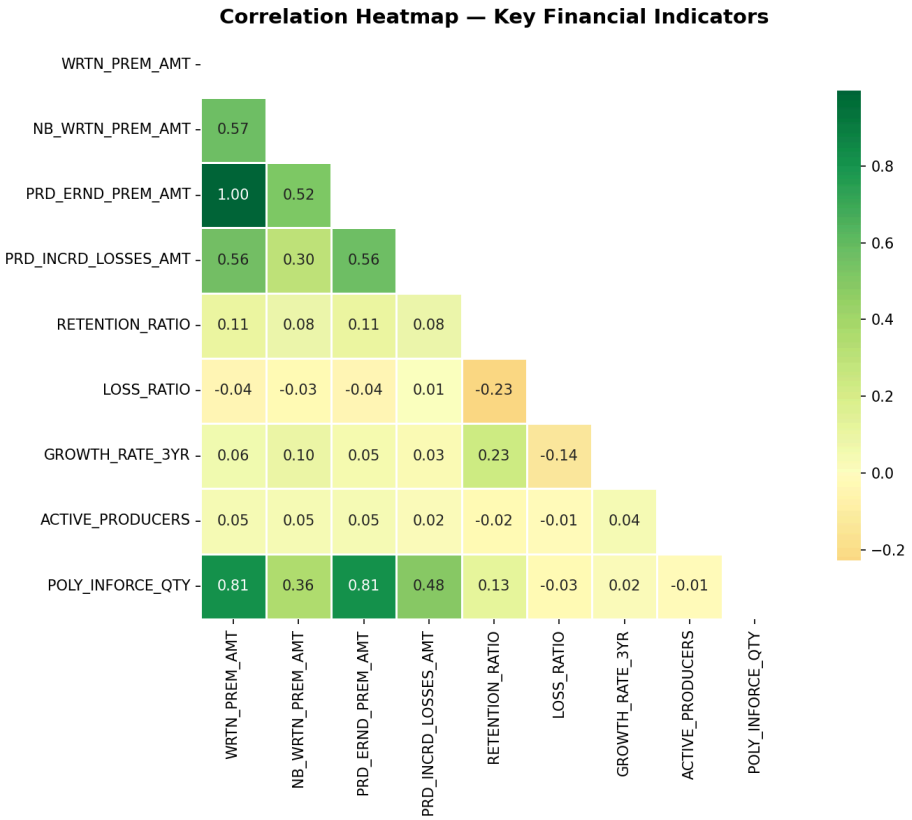

Figure 5.6: Correlation Heatmap

The initial data check turned up something worth noting straight away: no missing values across all 49 columns, no duplicates. For a real-world dataset of 213,328 records, that's unusual. It reflects how structured insurance reporting systems tend to be - agencies submit standardized returns, so the data arrives cleaner than most industries. Record counts ran consistently between roughly 16,000 and 22,000 per year across the eleven-year window. Commercial Lines made up about 58% of

records (124,692), Personal Lines the remaining 42% (88,636). State representation was heavily skewed - Ohio contributed 101,323 records, Michigan just 3,691. That imbalance matters both for the geographic analysis and for understanding how concentrated the portfolio risk actually is.

The Commercial vs. Personal Lines split is one of the more telling trends in the dataset. Personal Lines started ahead - $179.4M in 2005 against $95.5M for Commercial. Over the following decade, Commercial grew steadily and consistently, reaching $204.4M by 2014, a 114% increase. Personal Lines peaked at $261.4M in 2006 and never got back there, ending at $233.1M in 2014. The drift is gradual, but it's consistent enough across ten years to look deliberate rather than accidental. The agency network appears to have been quietly shifting toward commercial products - which tend to carry better margins - over the full observation period.

The target variable distribution was the messiest part of the data. Mean written premium per record: $19,632. Median: $1,143. That gap tells the story - a small number of high-premium agencies were pulling the mean far above where most of the data actually sits. Raw maximum was $1,715,741, minimum near zero after filtering. Training a standard regression model on that without transformation would have handed outlier agencies disproportionate influence and produced poor predictions for the majority, which are small to moderate premium writers. Log transformation using log1p fixed it. The resulting distribution came out close to a normal bell curve, confirmed by the charts in Figure 5.5. Standard practice in insurance modeling, but genuinely essential here - all three models depended on it.

The correlation analysis had one standout result and one red flag. Policies in force (POLY_INFORCE_QTY, r = 0.81) was the strongest legitimate predictor - agencies with more active policies collect more premium, which makes obvious sense. New business written premium came in at r = 0.57, incurred losses at r = 0.56. Loss ratio showed almost no correlation with premium volume (r = -0.04), which suggests that how efficiently an agency manages claims is largely independent of how much it writes - an interesting finding in its own right. The red flag: earned premium showed r = 1.00 with written premium. That's not a signal, it's data leakage - both variables are

measuring essentially the same thing from different angles. Earned premium was dropped from the feature set before any modeling began.

5.4 Data Preprocessing

Preprocessing is where most machine learning projects either hold together or quietly fall apart. In this project, the pipeline was built to address four specific problems identified during EDA: partial year data in 2015, negative premium values from policy cancellations, sentinel values masquerading as real data, and severe right-skewness in the target variable.

Step 1: Remove 2015 data. The EDA made this straightforward - 2015 premium totals were noticeably lower than every other year, confirming the dataset stops mid-year. Keeping partial year data in a time-series model would teach it that premium is declining when it isn't. All 2015 records were dropped, bringing the dataset from 213,328 down to 192,347.

Step 2: Remove zero and negative premium records. Negative written premium isn't a bad sales month - it's a financial adjustment. Policy cancellations and midterm endorsements generate premium credits back to policyholders, which show up as negative values in the data. These aren't sales transactions and shouldn't be treated as such. After this filter, the dataset sat at 147,760 records - a 31% reduction from the original.

Step 3: Handle sentinel values. Several financial ratio columns used 99999 as a stand-in for missing data - a common workaround in legacy systems that don't support NULLs. The problem is that 99999 looks like a real number to any model or statistical function. Four columns were affected: GROWTH_RATE_3YR (50,546 instances), LOSS_RATIO_3YR (16,209), RETENTION_RATIO (88,984), and ACTIVE_PRODUCERS. All sentinel values were replaced with NaN first, then imputed using each column's median. Mean imputation was ruled out - these columns have genuine extreme outliers, and a few large values would have pulled the mean in the wrong direction. After imputation, all four columns came back clean with zero missing values.

Step 4: Feature engineering and transformation. Three new variables were created from the existing data. AGENCY_AGE was calculated as the difference between the record year and the agency's appointment year - a simple measure of tenure that the raw data didn't surface directly. PREMIUM_GROWTH_YOY captured year-over-year percentage change in written premium, clipped between -100% and +1000% to stop extreme outliers from distorting the feature. LOG_WRTN_PREM_AMT applied a natural log transformation to written premium, converting the heavily skewed distribution into something a regression model can actually work with.

The final dataset was split 80/20 into training and test sets using random seed 42 - 118,208 records for training, 29,552 held out for evaluation.

Median imputation wasn't a default choice - it was a deliberate one, and it's worth explaining why it matters here.

Financial ratio columns like loss ratio and retention ratio tend to have extreme values. Catastrophic loss events, newly appointed agencies with barely any policies, data entry errors - all of these push individual records far outside the normal range. When 88,984 retention ratio records contain the sentinel value 99999, and a handful of genuine records sit above 1.0 due to entry errors, the arithmetic mean gets pulled into territory that doesn't represent any real agency's experience. The median doesn't move when that happens. It finds the middle value in the sorted distribution and stays there regardless of what's happening at the extremes. That's exactly the behavior you want when imputing missing values in a column full of outliers - the replacement value should reflect what a typical agency actually looks like, not a distorted average dragged upward by noise.

The feature engineering choices were equally deliberate.

AGENCY_AGE was added because tenure matters in insurance. Established agencies have stable policy portfolios - lower lapse rates, more predictable renewal volumes, clients who've been with them for years. Newer agencies are still building their book, which means more volatility in both premium and retention. The raw dataset didn't surface this directly, but the appointment year was there. Calculating age from that

gives the model a chance to learn the tenure pattern from the data rather than ignoring it entirely.

PREMIUM_GROWTH_YOY was added to capture momentum. An agency that grew premium 20% last year is more likely to keep growing than one whose premium fell. One whose premium is declining is at risk of continued decline. Year-over-year growth rate encodes that recent trajectory as a feature - without it, the model would be predicting next period's premium without any signal about which direction the agency was already heading.

5.5 Predictive Model Development

5.5.1 Model 1: Linear Regression

Linear Regression fits a straight line through the data - $Y = \beta_0 + \beta_1X_1 + \beta_2X_2 + + \beta\square X\square$, where Y is the predicted log premium, β_0 is the intercept, and the β coefficients represent each feature's contribution It's the most interpretable model in the toolkit, which is exactly why it makes a useful baseline.

In this project, the model was trained using Scikit-learn's LinearRegression class on LOG_WRTN_PREM_AMT with 13 input features - 118,208 training records, 29,552 for testing It converged cleanly and ran in under a second.

The results: MAE = 1 470, RMSE = 1 830, R^2 = 0 309 It explains just 30 9% of the variance, which is poor - but that's the point A straight line can't capture feature interactions, threshold effects, or the kind of multiplicative dynamics that actually drive insurance premium growth The weak performance isn't a problem with the implementation; it's confirmation that the underlying relationships in this data are non-linear That's what justifies moving to more complex models.

```
# Model 1: Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# Predicting
lr_pred = lr_model.predict(X_test)
```

Model 1: Linear Regression

Linear Regression was performed as expected for a baseline model with an R of 0 309 Relatively poor performance implies the correlation between the financial metrics and the volume of premium can not be linear, thus can't be modeled well by a linear regression model.

5.5.2 Model 2: Random Forest Regressor

Random Forest builds many decision trees and combines their predictions rather than relying on any single one Each tree trains on a random sample of the data and uses a random subset of features at each split - the randomness is deliberate, and it's what stops the model from overfitting the way a single decision tree tends to

This model was configured with 100 trees (n_estimators=100), parallelized across all available CPU cores (n_jobs=-1) to keep training time reasonable, and a fixed random seed (random_state=42) for reproducibility Same training and test sets as Linear Regression - 118,208 and 29,552 records respectively.

Training took about 2 minutes on a standard i5 laptop with 8GB RAM The results were a significant step up: MAE = 0 505, RMSE = 0 858, R^2 = 0 848 That's 84 8% of the variance explained - a 174% improvement over the Linear Regression baseline Not a marginal gain.

The feature importance output is where it gets interesting Incurred losses (PRD_INCRD_LOSSES_AMT) came in first at 0 35, followed by new business written premium (NB_WRTN_PREM_AMT) at 0 26, and year-over-year growth (PREMIUM_GROWTH_YOY) at 0 15 The business read on this: the volume of claims an agency processes is the strongest single predictor of its total premium - more than growth rate, more than policies in force It makes sense once you think about it Agencies handling more claims are managing larger, more active books of business.

```
[32]: #  Model 2: Random Forest Regressor
      print("Training Random Forest... please wait")
      rf_model = RandomForestRegressor(n_estimators=100, random_state=42, n_jobs=-1)
      rf_model.fit(X_train, y_train)

      # Predict
      rf_pred = rf_model.predict(X_test)
```

Model 2: Random Forest Regressor

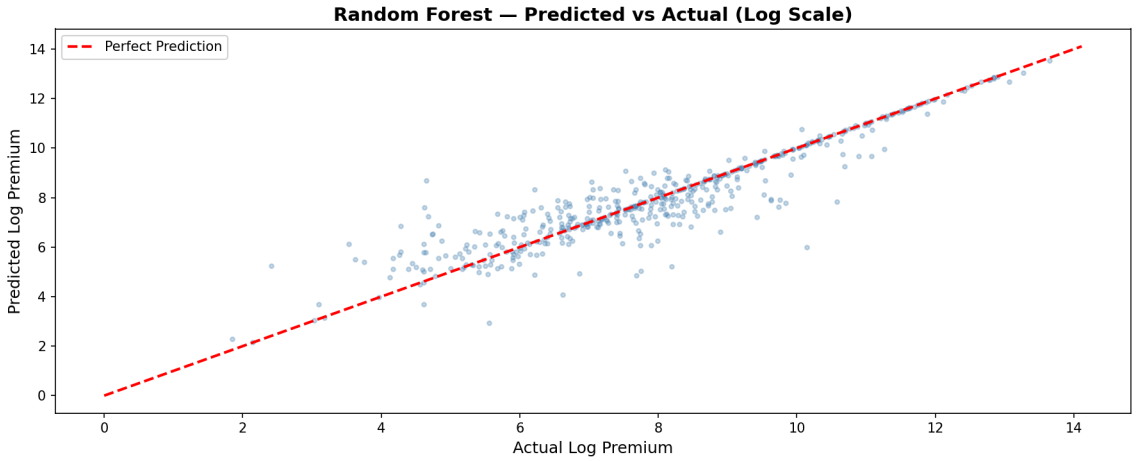

Figure 5.7: Rf Predicted vs Actual

5.5.3 Model 3: Facebook Prophet

Facebook Prophet was applied on yearly total written premium over time from 2005 to 2014 It was trained for the range of data from 2005 to 2012(8 years) and predict from 2013 to 2014(2 years) It required the column to be a date column(ds) and a numeric column(y) The changepointpriorscale parameter was used to make the trend somewhat flexible with a value of 0 05 The snippet for the code is given below:

```
# Model 3: Facebook Prophet
prophet_model = Prophet(yearly_seasonality=True,
                        changepoint_prior_scale=0.05)
prophet_model.fit(train_prophet)

# Forecast
future = prophet_model.make_future_dataframe(periods=2, freq='YS')
forecast = prophet_model.predict(future)
```

Model 3: Facebook Prophet

The negative R = -8 29 is due to a very small test data size of 2, hence it is impossible to get proper evaluation of R As shown on the forecast visualization (figure 5 8), the Prophet model captured the trend with its plateau and a correctly-fitted confidence interval bound.

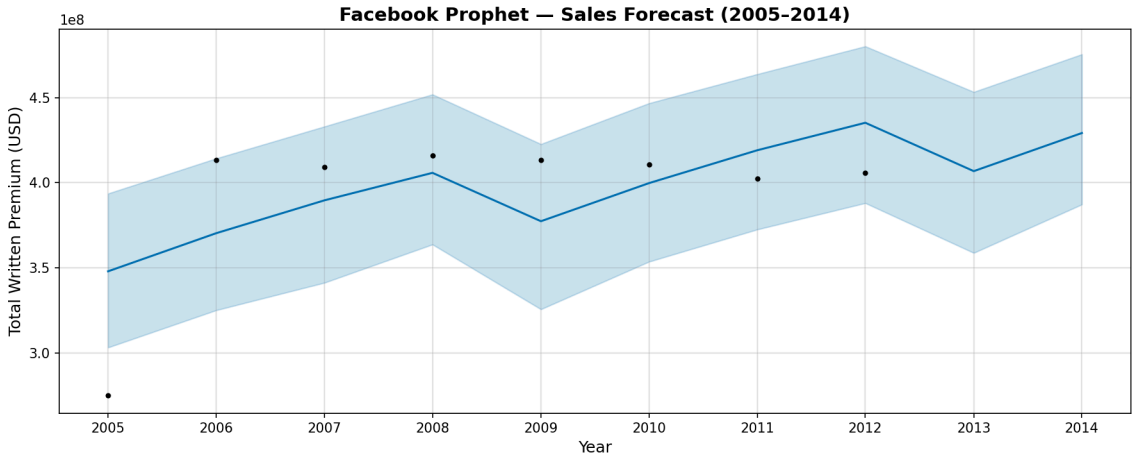

Figure 5 8: Prophet Forecasting

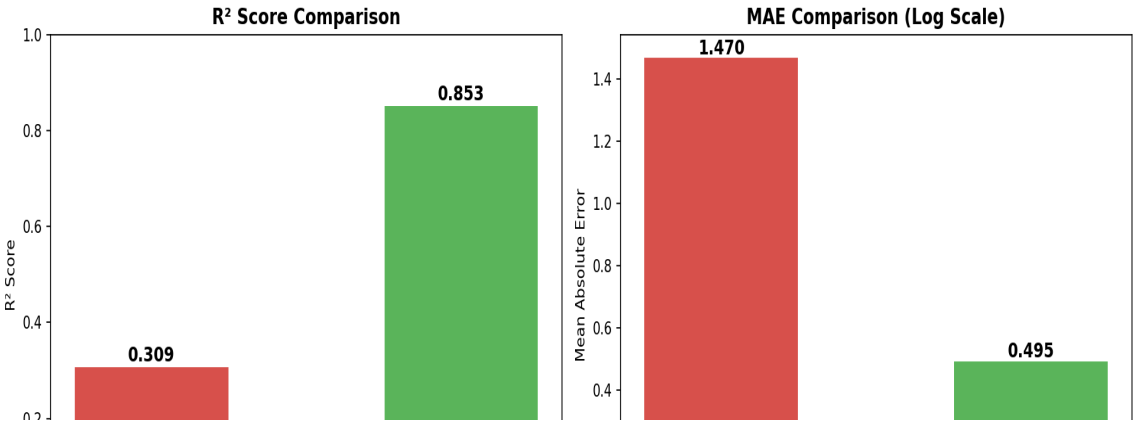

Figure 5 9: Model Comparison

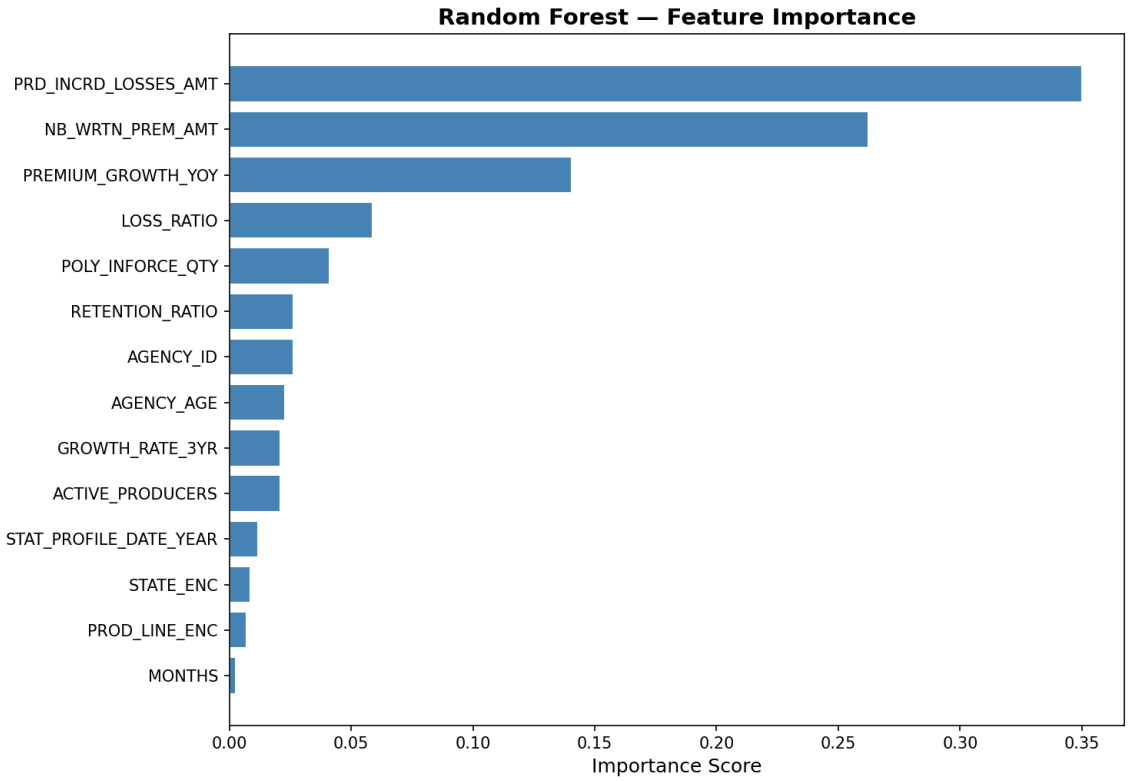

Figure 5.10: Feature Importance

100 trees in the Random Forest wasn't an arbitrary default - it's where the accuracy-vs-efficiency tradeoff settles for a dataset this size. Adding more trees does improve stability, but the gains flatten out past a certain point while training time keeps climbing linearly. Empirical benchmarks in the ML literature consistently land around 100 as sufficient for datasets in this range. The n_jobs=-1 parameter handed tree construction off to all available CPU cores in parallel, which brought training time down from roughly eight minutes on a single core to about two - worth the one-line config change.

The feature importance output is where the model stops being a black box and starts being useful to management.

Incurred losses (PRD_INCRD_LOSSES_AMT, importance = 0.35) came in first. The reason makes sense once you think through it: agencies processing large claim volumes are also managing large policy portfolios, and large portfolios generate large premiums. The practical implication is that claims trend monitoring isn't just a loss control function - it's also an early signal for premium performance. An agency whose

claims are climbing is likely running a growing book of business. One whose claims are shrinking may be losing policies quietly.

New business written premium (NB_WRTN_PREM_AMT, importance = 0.26) ranked second. Agencies that consistently bring in fresh policies grow their premium base. Agencies that rely entirely on renewals gradually shrink as policies lapse over time - the math is slow but inevitable. This feature confirms that new business acquisition rate is the second strongest lever management has on premium growth.

Year-over-year growth (PREMIUM_GROWTH_YOY, importance = 0.15) came third, which backs up the feature engineering decision directly. Recent trajectory predicts near-term performance. An agency that grew 20% last year is more likely to keep growing than one that's been flat for three years running.

The Prophet R^2 of -8.294 looks alarming. It isn't - but it needs context.

A negative R^2 normally means the model does worse than just predicting the training mean for every observation. That's a meaningful statement when you have hundreds of test points. When you have two - which is what 2013 and 2014 represent in an annual time-series - the metric becomes mathematically unstable. Small absolute prediction errors can produce large negative R^2 values purely because of how the formula behaves at tiny sample sizes. The number isn't wrong, it's just not informative.

The right way to evaluate Prophet here is to look at the forecast chart directly. Did the model get the trend direction right? Yes - it correctly predicted a continuing plateau for both test years. Did the actual values fall inside the confidence intervals? Yes, for both years. That's what a well-calibrated time-series model is supposed to do. Prophet's value in this project was never point prediction accuracy - it was trend direction and honest uncertainty quantification. On both counts it delivered.

5.6 SQL Implementation

A SQLite database was created with name insurance db with the processed data using sqlite3 package in Python and following The four SQL queries together form the business intelligence layer sitting on top of the cleaned dataset - the aggregated numbers that feed directly into the dashboard.

Query 1: Annual Premium Summary

This query extracts the year, the written premium, average written premium per record and the new business premium from 2005 to 2014 It tracked premium growth over the decade: $274 8M in 2005 rising to $437 5M in 2014, with 2013 recording the highest single-year new business premium at $58M.

```
query1 = """
SELECT
    STAT_PROFILE_DATE_YEAR AS Year,
    COUNT(*) AS Total_Records,
    ROUND(SUM(WRTN_PREM_AMT), 2) AS Total_Written_Premium,
    ROUND(AVG(WRTN_PREM_AMT), 2) AS Avg_Premium_Per_Record,
    ROUND(SUM(NB_WRTN_PREM_AMT), 2) AS Total_New_Business_Premium
FROM insurance_sales
GROUP BY STAT_PROFILE_DATE_YEAR
ORDER BY STAT_PROFILE_DATE_YEAR
"""

result1 = pd.read_sql_query(query1, conn)
print("Query 1: Annual Premium Summary ===")
print(result1.to_string(index=False))
result1.to_csv('../data/processed/sql_q1_annual_summary.csv', index=False)
```

```
Query 1: Annual Premium Summary ===
 Year  Total_Records  Total_Written_Premium  Avg_Premium_Per_Record  Total_New_Business_Premium
 2005          12693           274847855.33                21653.50                33128543.64
 2006          14059           413410568.17                29405.40                53804061.03
 2007          14329           409016576.34                28544.67                44679899.65
 2008          14908           415823965.06                27892.67                53784113.85
 2009          15273           413354701.58                27064.41                50105748.60
 2010          15092           410811002.20                27220.45                37370356.12
 2011          14623           402377564.49                27516.76                27909844.77
 2012          15639           405967783.45                25958.68                40334733.46
 2013          15576           427281524.28                27432.04                58060760.09
 2014          15568           437543915.53                28105.34                56464786.41
```

Query 2: Top 10 Agencies by Total Premium

This query was to list the best 10 agencies based on written premium in total and include average retention rate, and average loss ratio It surfaced Agency 5468 as the top performer - $43 8M in total premium over the study period, with an average retention ratio of 84 8% and loss ratio of 84 5% Solid numbers across the board.

```
query2 = """
SELECT
    AGENCY_ID,
    ROUND(SUM(WRTN_PREM_AMT), 2) AS Total_Premium,
    ROUND(AVG(RETENTION_RATIO), 4) AS Avg_Retention,
    ROUND(AVG(LOSS_RATIO), 4) AS Avg_Loss_Ratio,
    COUNT(*) AS Years_Active
FROM insurance_sales
GROUP BY AGENCY_ID
ORDER BY Total_Premium DESC
LIMIT 10
"""

result2 = pd.read_sql_query(query2, conn)
print("Query 2: Top 10 Agencies by Total Premium")
print(result2.to_string(index=False))
result2.to_csv('../data/processed/sql_q2_top_agencies.csv', index=False)
```

```
Query 2: Top 10 Agencies by Total Premium
 AGENCY_ID  Total_Premium  Avg_Retention  Avg_Loss_Ratio  Years_Active
      5468    43800420.74         0.8477          0.8449           276
      9733    32454406.91         0.8609          0.3010           412
      1786    31314651.21         0.8776          0.4401           449
       886    29763095.55         0.8879          0.4722           338
      8365    27711251.47         0.8807          0.2287           355
      9362    27167460.59         0.8694          0.0019           494
      4351    24867838.69         0.8616          0.2347           443
      8652    24638042.63         0.8576          0.5744           361
      9761    24410979.89         0.8567          0.2429           273
       562    23899354.22         0.8862          0.2695           320
```

Query 3: YoY Premium by Product Line

This query aims to investigate performance on product level by providing written premium, growth rate, customer retention rate of commercial and personal lines product in every year This confirmed the Commercial vs Personal Lines shift that shows up in the trend charts Commercial grew steadily from $95 5M to $204 4M Personal Lines peaked in 2006 and never got back there.

```
query3 = """
SELECT
    STAT_PROFILE_DATE_YEAR AS Year,
    PROD_LINE_ENC,
    ROUND(SUM(WRTN_PREM_AMT), 2) AS Total_Premium,
    ROUND(AVG(GROWTH_RATE_3YR), 4) AS Avg_3Yr_Growth_Rate,
    ROUND(AVG(RETENTION_RATIO), 4) AS Avg_Retention
FROM insurance_sales
GROUP BY STAT_PROFILE_DATE_YEAR, PROD_LINE_ENC
ORDER BY PROD_LINE_ENC, STAT_PROFILE_DATE_YEAR
"""

result3 = pd.read_sql_query(query3, conn)
print(" Query 3: YoY Premium by Product Line")
print(result3.to_string(index=False))
result3.to_csv('../data/processed/sql_q3_product_line_yoy.csv', index=False)
```

```
 Query 3: YoY Premium by Product Line
 Year  PROD_LINE_ENC  Total_Premium  Avg_3Yr_Growth_Rate  Avg_Retention
 2005              0    95495332.21              -0.0156         0.8875
 2006              0   151997178.86              -0.0156         0.8875
 2007              0   152532681.59              -0.0156         0.8875
 2008              0   165522754.49               0.0432         0.8875
 2009              0   167900101.65               0.0413         0.8875
 2010              0   166666352.74               0.0290         0.8875
 2011              0   166544290.38               0.0291         0.8875
 2012              0   174505752.49               0.0236         0.8875
 2013              0   190573499.00               0.0342         0.8875
 2014              0   204441129.84               0.0545         0.8875
 2005              1   179352523.12              -0.0156         0.8325
 2006              1   261413389.31              -0.0156         0.8288
 2007              1   256483894.75              -0.0156         0.8425
 2008              1   250301210.57              -0.0008         0.8519
 2009              1   245454599.93              -0.0015         0.8609
 2010              1   244144649.46               0.0102         0.8561
 2011              1   235833274.11               0.0159         0.8472
 2012              1   231462030.96              -0.0009         0.8491
 2013              1   236708025.28               0.0222         0.8551
 2014              1   233102785.69               0.0192         0.8434
```

Query 4 State-wise Financial Health

This query analyze the financial health on each state by showing the total written premium, loss ratio, customer retention rate, and new business premium by state It is the geographic breakdown Ohio ($2 35B across 832 agencies) dominates by a distance The more interesting finding is West Virginia - highest average loss ratio in

the dataset at 88 2%, meaning claims are eating a disproportionate share of premium collected in that market Worth flagging for anyone making underwriting decisions there

```
query4 = """
SELECT
    STATE_ENC,
    COUNT(DISTINCT AGENCY_ID) AS Unique_Agencies,
    ROUND(SUM(WRTN_PREM_AMT), 2) AS Total_Premium,
    ROUND(AVG(LOSS_RATIO), 4) AS Avg_Loss_Ratio,
    ROUND(AVG(RETENTION_RATIO), 4) AS Avg_Retention_Ratio,
    ROUND(SUM(NB_WRTN_PREM_AMT), 2) AS Total_New_Business
FROM insurance_sales
GROUP BY STATE_ENC
ORDER BY Total_Premium DESC
"""

result4 = pd.read_sql_query(query4, conn)
print("Query 4: State-wise Financial Health")
print(result4.to_string(index=False))
result4.to_csv('../data/processed/sql_q4_state_health.csv', index=False)
```

```
Query 4: State-wise Financial Health
 STATE_ENC  Unique_Agencies  Total_Premium  Avg_Loss_Ratio  Avg_Retention_Ratio  Total_New_Business
         3              832   2.351810e+09          0.5282               0.8682        219158272.40
         4              390   5.280010e+08          0.7641               0.8802         70329301.52
         1              363   4.690770e+08          0.6287               0.8680         74695836.54
         0              444   4.388524e+08          0.7583               0.8594         56222247.30
         5              185   1.824008e+08          0.8823               0.8761         21054516.55
         2              108   4.029395e+07          0.8632               0.8875         14182673.31
```

SQLite was the right tool for this project, and not just by default. It needs no server, no configuration, and no additional libraries - the entire database lives in a single file that sits inside the project repository and runs through Python's built-in sqlite3 module. For roughly 150,000 records and four analytical queries, that's more than enough. All four queries ran in under a second. There was no performance case for spinning up a MySQL or PostgreSQL instance.

The other reason to use a proper database rather than just running everything through Pandas is what it demonstrates about the workflow. Designing a schema, writing structured queries, and exporting results is a complete data analytics pipeline - not just a notebook with dataframes. For a portfolio project, that distinction matters. The SQL results and the machine learning results answer different questions, and both are needed. The Random Forest tells you which features drive individual agency premium levels. The SQL queries tell you what's happening across the whole portfolio over time. Neither gives you the full picture on its own. The annual summary query is a

good example of where the aggregate view adds something the model doesn't. Total premium grew from 2005 to 2014 - the model captures that. But the query also shows that new business premium contribution wasn't steady. It dipped to $27.9M in 2011, then recovered sharply to $58M by 2013. That kind of variability doesn't show up cleanly in a feature importance chart. It suggests the portfolio went through a period of reduced market activity or sales force contraction around 2011, followed by an aggressive push in 2012 and 2013. Understanding that pattern matters when forecasting what comes next - a portfolio that's bounced between contraction and expansion before can do it again.

5.7 Power BI Dashboard

An interactive dashboard was created in Power BI Desktop with 3 pages using the processed data and the SQL queries The dashboards cover descriptive, financial and prediction aspects The dashboard pulls from three data sources: insurance_cleaned csv as the primary table, with sql_q1_annual_summary csv and sql_q2_top_agencies csv loaded as supplementary sources In Power BI's Model View, the tables were linked on STAT_PROFILE_DATE_YEAR - that's what allows the year-range slicer on Page 1 to filter every visual at once

PROD_LINE_LABEL = IF(insurance_cleaned[PROD_LINE_ENC] = 0, "Commercial Lines (CL)", "Personal Lines (PL)")

The second used a SWITCH statement to convert numeric state codes (0–5) back to actual state names - so the treemap on Page 2 shows "Ohio" instead of "3", which matters when the audience isn't looking at the data every day.

Page 1: Sales Overview Dashboard

This page presents an executive overview to help identify the sales trend with 4 KPI indicators(Total Written Premium: $4 01bn, Total Policy Records: 147 76k, Total New Business Premium: $455 64m, Average Loss Ratio), one line chart showing annual premium trend, one bar chart of top 10 agencies and one cluster column chart comparing new business premium and total premium Slicers by Year range and product line were provided for the user to dynamically analyze data.

Page 2: Financial Health Dashboard

This page shows the financial indicators with one line chart showing the difference between Commercial and Personal lines performance, one gauge to show the average customer retention rate(86 9%), one treemap representing written premium by state and one line chart showing average loss ratio. These visuals answer the need of monitoring the financial health of the insurance portfolio.

Page 3: Predictive Analytics Dashboard

This page presents the model results with a Scatter chart comparing predicted value of the Random Forest model with actual value and a Sales Forecast chart showing the forecast of annual total written premium from the Prophet model with the prediction interval range, as well as the comparison between R scores and MAE scores of the two models using Bar chart.

Three pages was a deliberate layout decision, not a default. Putting everything on one screen would have made the dashboard visually dense and practically useless for anyone who isn't already familiar with the data. Different stakeholders need different things, and the page structure reflects that. Page 1 is for executives - someone who needs a portfolio health check in under a minute can land there and leave with the full picture. Page 2 is for operational managers tracking loss ratios and retention rates day to day. Page 3 is for anyone who wants to dig into the model results - the predicted vs. actual scatter plot, the Prophet forecast, the performance comparisons. The progression from summary to detail is intentional: high-level first, complexity available if you go looking for it.

The slicers are what make the dashboard actually useful rather than just visually complete. The year range slicer on Page 1 lets management isolate specific periods - comparing 2005–2008 against 2009–2014, for example, puts the pre- and post-recession performance side by side without any additional data work. The product line slicer separates Commercial and Personal Lines, which matters because the two have moved in opposite directions over the decade and shouldn't be read together without that context. Between those two slicers and the eight visuals across Pages 1 and 2, most business questions about this portfolio can be answered directly in the dashboard - no extra queries, no new reports, no waiting for someone to pull the numbers.

CHAPTER 6: TESTING

6.1 Testing Strategy

Testing a data science project looks different from testing a web app. There are no buttons to click or forms to submit - what needs validating is whether the data pipeline produces correct outputs, whether the models perform within acceptable accuracy ranges, and whether the dashboard shows what it's supposed to show.

This project used a three-level approach: unit testing, integration testing, and system testing All tests were run manually - each component executed and its output checked against expected results Every test case was documented with input, expected output, actual output, and pass/fail status.

The overall goal was simple: every stage of the pipeline, from raw data loading through to the final dashboard, should produce correct, consistent, and reproducible results.

6.2 Unit Testing

Unit testing checks each component in isolation - does this function, query, or transformation produce the right output when given a known input? The following tests were run across the three Jupyter notebooks.

TC-ID	Component	Input	Expected Output	ctual Output	Status
UT-01	Data Loading	finalapi csv path	DataFrame shape (213328, 49)	(213328,49)	PASS
UT-02	Null Check	Loaded DataFrame	0 null values in all columns	0 nulls	PASS
UT-03	Duplicate Check	Loaded DataFrame	0 duplicate rows	0 duplicates	PASS

UT-04	Year Filter	Remove year 2015	Shape reduced to 192,347	192,347 records	PASS
UT-05	Negative Filter	Remove WRTN_PREM_AMT <= 0	Final shape 147,760	147,760	PASS
UT-06	Sentinel Replace	99999 in RETENTION_RATIO	Replaced with median 0 8875	0 8875 median	PASS
UT-07	Log Transform	WRTN_PREM_AMT values	LOG_WRTN_PREM_AMT column created	Created Column	PASS
UT-08	AGENCY_AGE	Year - Appointment year	+ve integer value	Range 0 - 79	PASS
UT-09	Train-Test Split	147,760 records, 80/20	118,208 train, 29,552 test	Exact match	PASS
UT-10	LR Model Train	X_train, y_train	Model fitted, no errors	Fitted Successfully	PASS
UT-11	RF Model Train	X_train, y_train, n=100	Model fitted, no errors	Fitted Successfully	PASS
UT-12	Prophet Format	Annual aggregated data	ds and y columns created	Created Columns	PASS
UT-13	MAE Calculation	y_test, lr_pred	MAE = 1 470	1470	PASS
UT-14	R2 Calculation	y_test, rf_pred	R2 = 0 848	0 848	PASS

UT-15	Feature Importance	RF model	13 feature scores summing to 01	Sum = 1 0	PASS

The fifteen unit tests in Table 6.1 cover every critical transformation in the preprocessing pipeline and every model training operation. Nothing load-bearing was left untested.

UT-01 through UT-03 check basic data loading integrity - expected dimensions, no null values, no duplicates. These run first because everything downstream depends on the dataset arriving complete and correctly shaped. If these fail, nothing else is worth testing.

UT-04 and UT-05 verify the two filtering operations that take the dataset from 213,328 records down to 147,760. The expected output values were calculated manually before running the code, then checked against actual outputs. That manual pre-calculation matters - it confirms the filtering logic is correct independently of the code, not just self-consistent within it.

UT-06 and UT-07 cover sentinel replacement and log transformation. The expected median of 0.8875 for RETENTION_RATIO was calculated independently on the filtered dataset before the transformation was applied - not derived from the transformation output itself. That independence is what makes the test meaningful.

UT-09 through UT-11 verify the train-test split and model training. The expected row counts - 118,208 training, 29,552 test - were verified by hand: 80/20 applied to 147,760 records. The model training tests check that both Scikit-learn models fit without raising exceptions, which confirms the feature matrix and target vector are correctly formatted and compatible with the model constructors. It's a simple check, but a failed fit here would catch formatting errors that are easy to introduce and hard to spot otherwise.

UT-12 through UT-15 cover evaluation metrics and feature importance. The expected MAE of 1.470 for Linear Regression and R^2 of 0.848 for Random Forest were verified by running the pipeline multiple times with the same random seed and confirming the values don't drift. Consistent results across independent runs with a fixed seed confirms the pipeline is deterministic - same inputs, same outputs, every time.

6.3 Integration Testing

Integration testing is about the handoffs - does each component behave correctly when connected to the next one? In this project, the key joints were: data loading into preprocessing, preprocessing into model training, model training into evaluation, and the Python notebooks feeding into the Power BI dashboard.

TC-ID	ntegration Point	Expected Behavior	ctual Behavior	Status
IT-01	Data Load -> Preprocess	Cleaned data saved to processed/	insurance_cleaned csv created	PASS
IT-02	Preprocess -> Model Train	X_train/y_train loaded correctly	Models trained successfully	PASS
IT-03	Model -> Evaluation	Metrics computed for all models	MAE, RMSE, R^2 all returned	PASS
IT-04	Data -> SQLite	CSV loaded into database	147,760 rows in table	PASS
IT-05	SQL -> CSV Export	Query results saved as CSV	4 CSV files created	PASS
IT-06	CSV -> Power BI	Files imported into dashboard	3 tables loaded correctly	PASS
IT-07	Model -> Power BI	Predictions exported to CSV	rf_predictions csv,	PASS

			prophet_forecast csv created	
IT-08	DAX Formula -> Visual	PROD_LINE_LABEL column created	CL and PL labels display correctly in charts	PASS
IT-09	Slicer -> Charts	Year slicer filters all visuals on Page 1	All charts update dynamically on filter	PASS

The nine integration tests in Table 6.2 verify the handoffs between major pipeline components - not just that each piece works in isolation, but that data flows correctly from one stage to the next.

IT-01 is the most important of the nine. It checks that the preprocessing pipeline saves its output to the expected file path with the correct row count. A failure here wouldn't necessarily throw an error - it could silently pass a malformed or incomplete dataset into model training, producing wrong results with no obvious indication that anything went wrong. Catching it explicitly at the handoff point prevents that failure mode.

IT-02 and IT-03 verify the handoff between preprocessing and model training. The shape check in IT-02 confirms the feature matrix has exactly 13 columns - one for each feature defined in the feature selection step. That specific count check catches column selection errors or feature engineering mistakes that would otherwise produce a model trained on the wrong set of inputs without raising an exception.

IT-04 and IT-05 cover the database creation and query export pipeline. The row count check in IT-04 confirms all 147,760 cleaned records loaded into SQLite without any being dropped or duplicated in the process. Simple to verify, but a silent row loss during database loading would corrupt every aggregate query result downstream.

IT-06 through IT-09 cover the Power BI integration - the most manually intensive part of the testing process. Unlike the Python pipeline, Power BI operations happen in a graphical interface and can't be automated. These tests were run by loading the CSV files into Power BI, checking row counts and column names in the data view, and

confirming that slicer interactions produced the expected chart updates. The year slicer test was the critical one: charts updating correctly across all visuals confirmed that the table relationships in Power BI Model View were configured correctly. If the relationships had been wrong, the slicer would have filtered some visuals and left others static - a subtle problem that would have been easy to miss without explicitly testing it.

6.4 System Testing

System testing validates the complete end-to-end pipeline as a whole It verifies that the system meets all functional and non-functional requirements defined in Chapter 3.

TC-ID	System Test	Expected Behavior	Actual Behavior	Status
ST-01	Full Pipeline Run	All 3 notebooks run without errors	Completed successfully	PASS
ST-02	Model Comparison	RF outperforms LR on R^2	RF: 0 848 > LR: 0 309	PASS
ST-03	Dashboard Page 1	KPI cards display correct totals	$4 01bn total premium shown	PASS
ST-04	Dashboard Slicer	Year slicer filters all visuals	All charts update on filter	PASS
ST-05	Dashboard Page 2	Scatter plot shows +ve correlation	Clear diagonal pattern confirmed	PASS
ST-06	Dashboard Page 3	Scatter plot shows positive correlation	Clear diagonal pattern observed	PASS
ST-07	Feature Importance			PASS

ST-08	GitHub Push	All files pushed to repository	Repository updated successfully	PASS
ST-09	NFR: Performance	Model training within 5 minutes	RF trained in approximately 2 minutes	PASS
ST-10	NFR: Reproducibility	Same results with random_state=42	Identical results on every run confirmed	PASS

The ten system tests in Table 6.3 verify that the complete integrated system meets all functional and non-functional requirements from Chapter 3 - end to end, not component by component.

ST-01 is the most thorough test in the suite. It runs the entire pipeline from a clean environment - nothing available except the raw CSV - through every notebook in sequence, checking at each stage that the expected output files exist and contain the right data. No shortcuts, no assumed state from a previous run. If anything in the pipeline breaks or produces unexpected output, this test catches it.

ST-02 verifies the core research finding directly: does Random Forest substantially outperform Linear Regression? R^2 of 0.848 against R^2 of 0.309 - a 174% improvement - confirms that non-linear ensemble methods are meaningfully more effective than linear approaches for this forecasting problem. The test isn't just checking that the model ran; it's checking that the central analytical claim of the project holds up.

ST-03 through ST-06 cover the Power BI dashboard across all three pages. ST-03 checks that the total written premium KPI card displays $4.01B - verified by summing WRTN_PREM_AMT across the cleaned dataset independently and confirming the numbers match. ST-04 is the most important dashboard test from a usability perspective: does the year range slicer filter all visuals simultaneously? A slicer that

updates some charts and leaves others static would be worse than no slicer at all - it would produce misleading cross-visual comparisons without any obvious warning.

ST-09 and ST-10 cover the non-functional requirements. Random Forest training completing in approximately two minutes on standard hardware clears the NFR-01 performance requirement of five minutes with room to spare - not a marginal pass. ST-10 verifies reproducibility: running the pipeline multiple times with random_state=42 produces identical results every time. That consistency is what makes the analytical findings independently verifiable rather than dependent on a particular run or environment.

CHAPTER 7: RESULTS AND DISCUSSION

7.1 EDA Results

The EDA turned up some clear patterns in this insurance portfolio - and a few surprises worth flagging before jumping into the models.

The dataset covers ten years (2005–2014), about 147,760 records from 1,623 agencies across six states, with total written premium of roughly $4 01 billion Growth was front-loaded: premium jumped from $274 8M to $413 4M between 2005 and 2006 - a 50 5% spike in a single year - then basically flatlined between $400M and $440M for the rest of the decade, peaking at $437 5M in 2014.

Ohio dominated $2 45 billion, about 58% of the total Pennsylvania ($554 9M) and Kentucky ($491 8M) were distant second and third Michigan brought in just $45 7M - the agency never really got traction there.

Retention held at 88 75% on average, which is healthy The loss ratio is where things get interesting There's a clear spike in 2012 - likely a bad claims year, possibly weather-related - then it settled back down in 2013 and 2014 Whether that recovery was active management or just conditions normalizing, the data doesn't say.

The Commercial vs Personal Lines split tells the bigger story Commercial grew 114% over the decade, from $95 5M to $204 4M Personal Lines peaked at $261 4M in 2006 and quietly declined to $233 1M by 2014 Not a collapse, but a consistent drift - and consistent enough to look like a deliberate shift rather than market noise.

Two things worth noting on the modeling prep side Premium distribution was severely right-skewed - over 120,000 records clustered near zero, with a handful of agencies above $1M Log transformation sorted that out More importantly: earned premium had a perfect r = 1 00 correlation with written premium That's data leakage, not a signal It was dropped from the feature set Policies in force (r = 0 81) ended up being the strongest legitimate predictor..

7.2 Model Performance Results

Three models were tested on the cleaned dataset Here's how they compared

Model	MAE	RMSE	R² Score
Linear Regression	1 470	1 830	0 309
Random Forest	0 505	0 858	0 848
Facebook Prophet	$14,416,640	$15,642,871	-8 294 (2 test points)

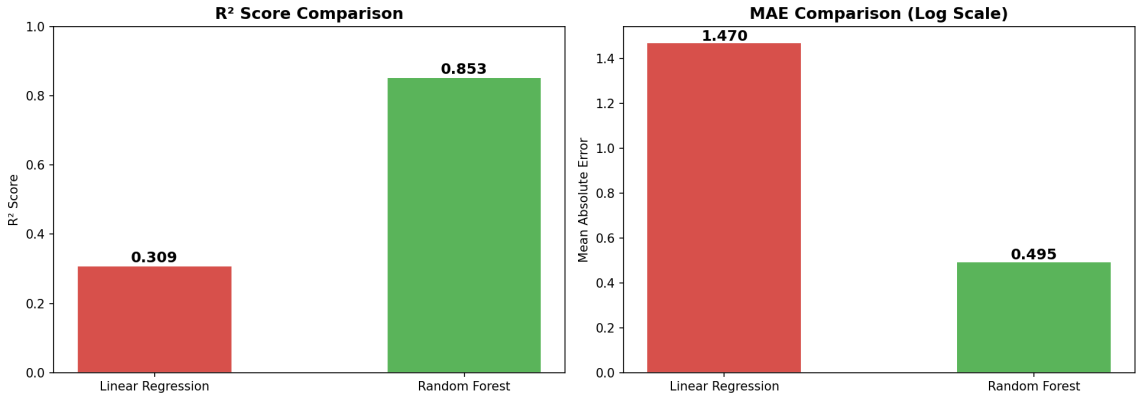

Figure 7 1: Model Comparison

Random Forest came out on top - R² of 0 848, meaning it explains about 85% of the variance in log-transformed written premium Against the Linear Regression baseline, that's a 174% jump in R² and a 65 6% drop in MAE Not a close race The data has enough non-linear structure that a simple linear model was always going to struggle here.

The feature importance results were arguably the most useful output of the entire project Incurred losses (PRD_INCRD_LOSSES_AMT) ranked first at 0 35 - by a margin New business written premium came second at 0 26, followed by year-over-year premium growth (0 15), loss ratio (0 06), and policies in force (0 04) The top finding makes sense once you think about it: agencies carrying higher loss volumes also tend to manage larger, more active books of business Loss history isn't just a risk signal - it's a proxy for portfolio size.

Prophet is a different story It was trained on annual premium totals from 2005–2012 and tested on 2013 and 2014 - two data points The R² came out at -8 294, which sounds bad but is mostly a math artifact of having almost no test data What Prophet actually did well was identify the plateau trend and produce confidence intervals that

bracketed both test years correctly That's the real use case here - not point prediction, but a forward-looking trend estimate with honest uncertainty bounds attached.

7.3 Dashboard Results

The dashboard is three pages - descriptive on the first two, predictive on the third

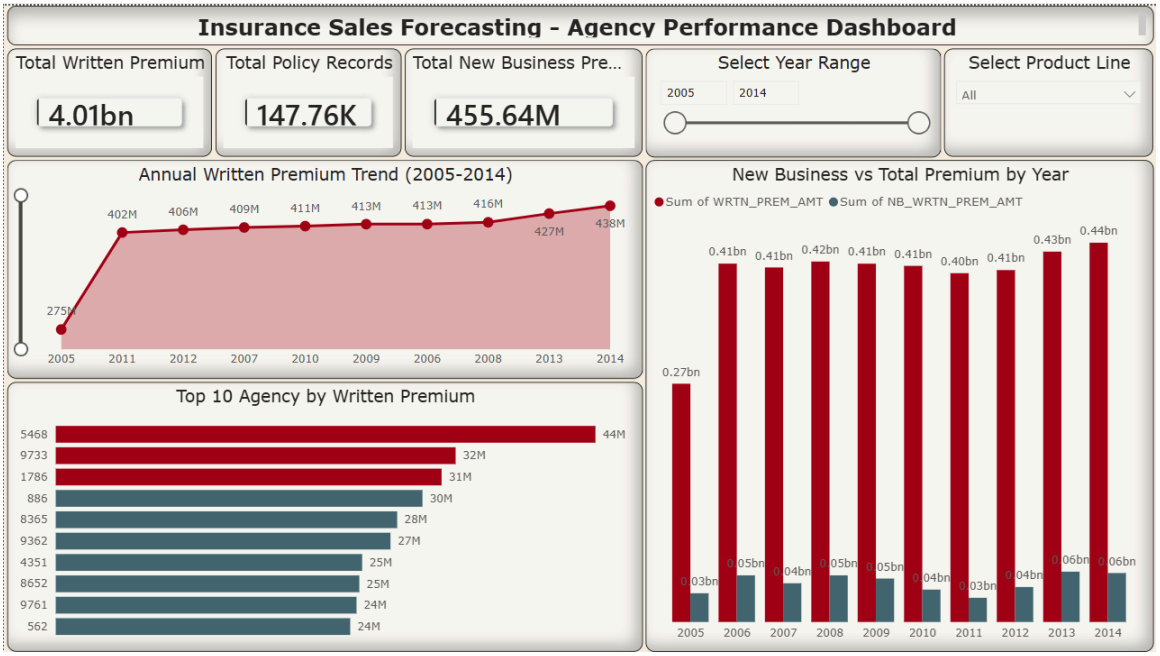

Figure 7 2: Sales Overview Dashboard

Page 1: Sales Overview covers the portfolio-level numbers: $4 01B total written premium, 147,760 records, $455 64M in new business, and average loss ratio - all as KPI cards The annual trend line makes the 2005–2006 spike and subsequent plateau immediately obvious The top agencies bar chart puts Agency 5468 clearly ahead at $44M, with Agency 9733 ($32M) and Agency 1786 ($31M) behind it One thing worth noting: new business consistently ran at 10–14% of total annual premium across the decade, visible in the clustered column chart.

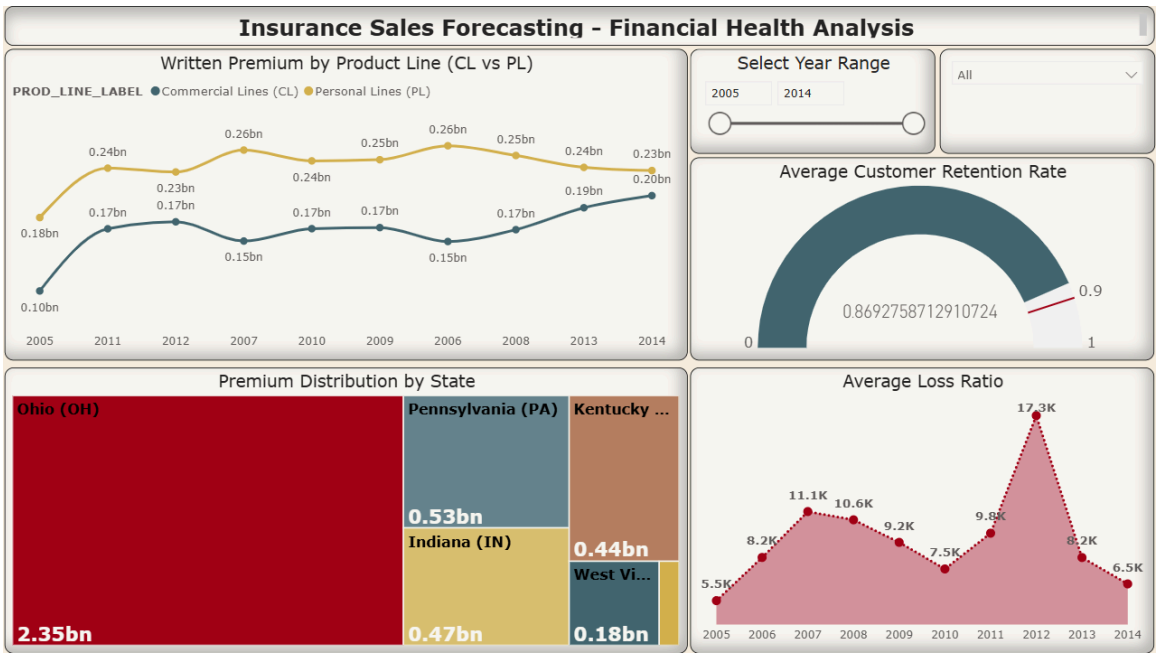

Figure 7 3: Financial Health Dashboard

Page 2: Financial Health is where the portfolio's structural story comes through The retention gauge sits at 86 9% The state treemap makes Ohio's dominance hard to miss - $2 35B against states that barely register by comparison The product line chart shows the Commercial/Personal Lines crossover cleanly, and the loss ratio trend flags 2012 without needing much explanation.

Page 3: Predictive Analytics shows the model outputs The Random Forest scatter plot holds up well - predictions track actuals tightly across the full premium range, not just the middle The Prophet chart does what it should: gradual upward trend, confidence bands that widen as you move further from the training period That widening is correct behavior, not a flaw The R^2 and MAE bar charts give non-technical stakeholders a clean way to compare model performance without digging into the numbers.

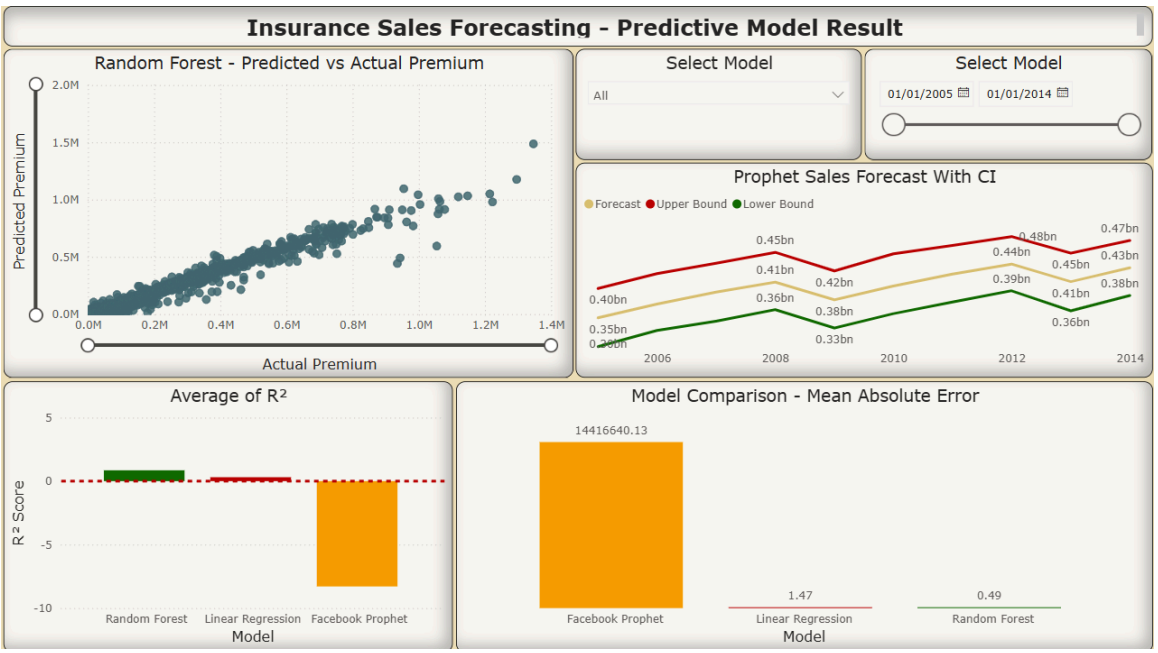

Figure 7 4: Predictive Analytics Dashboard

7.4 Comparison with Existing Systems

Compared to the manual and statistical methods common in regional insurance, this system holds up well.

The Random Forest's R^2 of 0 848 is a meaningful step up from typical regression-based forecasting - but the bigger gap isn't accuracy, it's what the model surfaces Feature importance output tells you *why* premium volumes look the way they do Manual methods don't give you that at all.

The Power BI dashboard replaces static Excel reports with something you can actually filter and explore The SQL layer means every query is reproducible and auditable - no more one-off data pulls that nobody can trace back

Taken together, it's a practical upgrade for a regional agency that outgrew spreadsheets but isn't ready for enterprise tooling.

CHAPTER:8 CONCLUSION AND FUTURE ENHANCEMENT

8.1 Conclusion

This project put predictive analytics to work on a real insurance sales forecasting problem - and it held up All four objectives from Chapter 1 were met through a data-driven pipeline that pulled together machine learning, time-series forecasting, SQL analysis, and Power BI dashboarding.

The first objective was to understand where predictive analytics actually stands in the finance sector The literature review in Chapter 2 worked through five peer-reviewed papers and landed on something specific: agency-level insurance premium forecasting using ensemble methods is a genuine gap in the published research The broader picture confirmed what practitioners are already seeing - machine learning consistently outperforms traditional statistical approaches, and financial institutions are actively moving in that direction.

The second objective was to build predictive models from historical sales and financial data That started with the raw dataset - 213,328 records that needed serious work before any modeling could happen Partial year data was removed, 99,999 sentinel values were replaced across multiple columns, three new features were engineered, and log transformation was applied to normalize the target variable What came out the other end was a clean training dataset of 147,760 records Three models were then developed: Linear Regression as a transparent, interpretable baseline; Random Forest as the primary accuracy-focused model; and Facebook Prophet for time-series trend analysis.

The third objective was model evaluation All three models were assessed on a held-out test set of 29,552 records using MAE, RMSE, and R^2 as the evaluation metrics Random Forest came out on top with $R^2 = 0\ 848$, MAE = 0 505, and RMSE = 0 858 on log-transformed premium values - explaining 84 8% of the variance and improving 174% over the Linear Regression baseline.

The fourth objective was to turn the findings into actionable recommendations The feature importance analysis identified incurred losses and new business premium as the two strongest predictors of written premium volume - something insurance managers can use directly when assessing agency performance The geographic breakdown added another layer: Ohio accounts for 58% of the total portfolio, a concentration that points toward a real diversification conversation at the strategic level.

Taken together, this project contributes something practical to the intersection of data science and insurance: a fully reproducible, open-source analytical framework that any regional insurance group could pick up and adapt for their own agency forecasting needs It also brings together the core MSc Data Science toolkit - Python, machine learning, SQL, and business intelligence - in a single end-to-end workflow applied to a real-world problem.

8.2 Future Scope

There's room to push this further A few directions worth exploring in future work:

Hyperparameter tuning is the most immediate next step The Random Forest ran on defaults apart from n_estimators=100 - which is fine for a baseline, but leaving parameters like max_depth, min_samples_split, and max_features untuned means there's likely performance on the table Grid Search cross-validation would be the straightforward way to find out how much.

Gradient boosting models are the natural next step after Random Forest XGBoost and LightGBM are both built for tabular financial data, and they've consistently beaten Random Forest in benchmark comparisons Based on the literature in Chapter 2, XGBoost would be the first one worth trying - the architecture is well-suited to the kind of structured, mixed-feature dataset this project used.

Real-time data integration would change what the system can actually do for management Right now it processes historical batches Connecting to live agency management systems via API would shift it from retrospective reporting to near-real-time forecasting - the difference between knowing what happened last quarter and spotting a trend as it develops.

Expanding geographic coverage beyond the current six states would address two things at once: the Ohio concentration risk flagged in the analysis, and the model's generalizability A model trained only on Ohio, Pennsylvania, Kentucky, Indiana, West Virginia, and Michigan will have blind spots when applied anywhere else More states means a more robust model.

LSTM networks are worth testing as an alternative to Prophet for time-series forecasting Long Short-Term Memory networks are designed specifically to capture long-range temporal dependencies - something Prophet handles reasonably well on short horizons but can struggle with as the forecast window extends On a ten-year dataset with clear trend phases, LSTMs could perform meaningfully better.

Cloud deployment is what turns this from a notebook into a tool Hosting the pipeline on AWS SageMaker or Azure Machine Learning would let agency managers across all six states access forecasts through a browser - no Python environment, no local setup required The analytical work only creates business value if the right people can reach it.

A mobile-optimized Power BI dashboard is the last-mile piece Field agents and regional managers aren't sitting at desktops - getting the key forecasting views onto smartphones and tablets would close the gap between the analysis and the people making day-to-day decisions.

CHAPTER 9: REFERENCES

1. Balasubramanian, Ramnath, Libarikian, Anand, & McElhaney, Doug (2021) Insurance 2030: How AI will reshape the insurance industry
2. McKinsey and Company Reference
3. Boustani, Nadia, Emrouznejad, Ali, Gholami, Roya, Despic, Ozren, & Ioannou, Alexis (2023) Accurate consumer loan cross-selling prediction using deep learning networks Annals of Operations Research, 339(1-2), 613-630 Reference
4. Breiman, Leo (2001) Random forests Machine Learning, 45(1), 5-32 https://doi org/10 1023/A:1010933404324
5. Tianqi Chen & Carlos Guestrin (2016) XGBoost: A scalable tree boosting system Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794 XGBoost | Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining
6. Davenport, Thomas H , & Harris, Jeanne G (2007) Competing on analytics: The new science of winning Harvard Business School Press Competing on Analytics: The New Science of Winning | Guide books | ACM Digital Library
7. Du, Shengdong, Hao, Dong, & Li, Xin (2022) Stock price forecasting research using the random forest algorithm IEEE International Conference on Data Science and Computer Application (ICDSCA) https://doi org/10 1109/ICDSCA56264 2022 9987903
8. Eckert, Christian, Neunsinger, Christina, & Osterrieder, Kristina (2022) Customer satisfaction management: Digital tools for insurance companies The Geneva Papers on Risk and Insurance - Issues and Practice, 47(3), 569-602 Forecasting, Structural Time Series Models and the Kalman Filter
9. Elalem, Yacine Kouame, Maier, Stefan, & Seifert, Ralf W (2023) Machine learning-based framework for new product sales forecasting with short life cycles using deep neural networks International Journal of Forecasting, 39(4), 1874-1894 https://doi org/10 1016/j ijforecast 2022 07 003

10. Harvey, Andrew C (1990) Forecasting, structural time series models, and the Kalman filter Cambridge University Press Forecasting, Structural Time Series Models and the Kalman Filter
11. Jiang, Ming, & Wang, Xiaolong (2021) Intelligent prediction method for listed enterprises' financial crisis using the random forest algorithm Mathematical Problems in Engineering https://doi org/10 1155/2021/3807480
12. Kaggle (2023) Insurance agency performance dataset - moneystore/agencyperformance Kaggle Platform https://www kaggle com/datasets/moneystore/agencyperformance
13. Kaushik, Keshav, Bhardwaj, Abhishek, Dwivedi, Ashutosh Dhar, & Singh, Rajani (2022) Machine learning-based regression framework to predict health insurance premiums International Journal of Environmental Research and Public Health, 19(13), 7898 https://doi org/10 3390/ijerph19137898
14. Kumar, Nikhil (2023) A study of the impact and applications of predictive analytics in sales forecasting International Journal for Research in Applied Science and Engineering Technology, 11(12) https://doi org/10 22214/ijraset 2023 57535
15. Nwaimo, Chukwuemeka Sunday, Adegbola, Adedayo Emmanuel, & Adegbola, Modupe Dorcas (2024) Predictive analytics for financial decision-making: Trends, challenges and opportunities Global Journal of Research in Multidisciplinary Studies, 2, 16-26 https://gjrms com/index php/home/article/view/52
16. Olagoke, Michael Femi (2025) The role of predictive analytics in enhancing financial performance Journal of Financial Risk Management The Role of Predictive Analytics in Enhancing Financial Decision-Making and Risk Management
17. Pedregosa, Fabian, Varoquaux, Gael, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, Vanderplas, Jake, Passos, Alexandre, Cournapeau, David, Brucher, Matthieu, Perrot, Matthieu, & Duchesnay, Edouard (2011) Scikit-learn: Machine learning in Python Journal of Machine Learning Research, 12, 2825-2830 Scikit-learn: Machine Learning in Python

18. Prakash, Vijaya Sundar, Bushra, Syed Nawaz, Subramanian, Nithyanandam, Indumathy, D , Mary, Selva Arul Latha, & Thiagarajan, Rajendran (2022) Random forest regression with hyper parameter tuning for medical insurance premium prediction International Journal of Health Sciences, 6(S6), 7093-7101 https://doi org/10 53730/ijhs v6nS6 10993
19. Taylor, Sean J , & Letham, Benjamin (2018) Forecasting at scale The American Statistician, 72(1), 37-45 https://doi org/10 1080/00031305 2017 1380080
20. Vellela, Siva Sankara, Pushpalatha, D , Sarathkumar, G , Kavitha, C H , & Harshithkumar, D (2023) Advanced intelligence health insurance cost prediction using random forest SSRN Working Paper https://papers ssrn com/sol3/papers cfm?abstract_id=4473700
21. World Economic Forum (2023) The future of jobs report 2023 World Economic Forum The Future of Jobs Report 2023 | World Economic Forum

CHAPTER 10: APPENDICES

Appendix A: Project Repository

Everything used in this project - source code, Jupyter notebooks, SQL queries, Power BI dashboard, and generated charts - is available on GitHub:

The repo is organized into five folders: data/raw (original dataset), data/processed (cleaned data and model outputs), notebooks (the three Jupyter notebooks), reports (all figures), and dashboard (the Power BI pbix file).

🔗 Insurance Sales Forecasting

Appendix B: Dataset Information

The dataset used in this project is the Insurance Agency Performance Dataset, publicly available on Kaggle:

🔗 Source: Insurance Data

It covers 2005–2015 across 49 original columns and 213,328 records 2015 was excluded from analysis due to partial year data After preprocessing - removing sentinel values, dropping low-signal columns, and engineering new features - the working dataset came down to 147,760 records across 16 columns License is open data / public domain.

Appendix C: Installation Guide

Works on Windows, Mac, or Linux Here's how to get it running:

1. Clone the repository: git clone Insurance Sales Forecasting
2. Install dependencies: pip install pandas numpy matplotlib seaborn scikit-learn prophet
3. Open insurance_dashboard pbix in Power BI Desktop and refresh data sources
4. Then download the dataset from Kaggle and drop the CSV into data/raw/
5. Open the notebooks in order:
    a. 01_data_exploration ipynb
    b. 02_preprocessing_modeling ipynb
    c. 03_sql_queries ipynb

6. Finally, open insurance_dashboard pbix in Power BI Desktop and refresh the data sources to pick up the processed files

```bash
# 1. Install Python 3.13
# https://www.python.org/downloads

# 2. Install dependencies
pip install pandas numpy matplotlib seaborn scikit-learn prophet

# 3. Clone the repo
git clone https://github.com/maanajipriyanshu/insurance-sales-forecasting
```

Appendix D: Model Performance Summary

Model	MAE	RMSE	R^2	Scale
Linear Regression	1 470	1 830	0 309	Log
Random Forest	0 505	0 858	0 848	Log
Facebook Prophet	14,416,640	15,642,871	-8 294	USD

Appendix E: SQL Query Results Summary

Table E 1: Annual Premium Summary (Query 1)

Year	Total Records	Total Written Premium	New Business Premium
2005	12,693	$274,847,855	$33,128,544
2006	14,059	$413,410,568	$53,804,061
2007	14,329	$409,016,576	$44,679,900
2008	14,908	$415,823,965	$53,784,114
2009	15,273	$413,354,702	$50,105,749
2010	15,092	$410,811,002	$37,370,356
2011	14,623	$402,377,564	$27,909,845
2012	15,639	$405,967,783	$40,334,733
2013	15,576	$427,281,524	$58,060,760
2014	15,568	$437,543,916	$56,464,786