Policy Cost

1 Introduction

In this assignment we will act as data scientists for a fictitious insurance company. our mission is to build a regression model that accurately predicts the Policy Cost based on a diverse set of anonymized customer and policy-related features. The dataset mimics real-world insurance data, making the task a comprehensive exercise in handling various data types and predicting continuous values.

2 Data

2.1 Dataset

The data files are as follows:

- **train.csv** - The primary dataset for training your machine learning models. It contains the features and the target variable (Policy Cost) for model learning.
- **x_test.csv** - This is the test set on which you will make predictions. It contains the same features as train.csv but *without* the Policy Cost column. Your final submission file should contain predictions for the Policy Cost for each ID in this file.
- **sample_submission.csv** - A sample submission file provided in the correct format. This file shows you exactly how your submission should be structured.

we use the train.csv to train the models.

2.2 Features

Our train data set contains features as follows:

The features are as follows:

```
 Column        Non-Null Count   Dtype
_________________________________

 0   ID              1100000 non-null  int64
 1   Years Lived       1082781 non-null  float64
 2   Sex             1100000 non-null  object
```

```
3   Yearly Earnings        1058849 non-null  float64
4   Relationship Status    1083018 non-null  object
5   Dependent Count        999297 non-null   float64
6   Academic Standing      1100000 non-null  object
7   Job Title              771458 non-null   object
8   Wellness Index         1031954 non-null  float64
9   Region                 1100000 non-null  object
10  Coverage Class         1100000 non-null  object
11  Prior Claims           766140 non-null   float64
12  Automobile Age         1099994 non-null  float64
13  Financial Rating       973595 non-null   float64
14  Coverage Period        1099999 non-null  float64
15  Coverage Commencement  1100000 non-null  object
16  Client Review          1028703 non-null  object
17  Tobacco Use            1100000 non-null  object
18  Physical Activity      1100000 non-null  object
19  Asset Category         1100000 non-null  object
20  Policy Cost            1100000 non-null  float64
```

Then I checked the percentage of missing values according to the whole data :

```
 Missing % in train:
 Years Lived            1.565364
Yearly Earnings         3.741000
Relationship Status     1.543818
Dependent Count         9.154818
Job Title              29.867455
Wellness Index          6.186000
```

```
Prior Claims         30.350909
Automobile Age         0.000545
Financial Rating      11.491364
Coverage Period        0.000091
Client Review          6.481545
dtype: float64
Missing % in test:
 Years Lived           1.486
Yearly Earnings        3.798
Relationship Status    1.547
Dependent Count        8.969
Job Title             29.533
Wellness Index         6.030
Prior Claims          30.169
Financial Rating      11.477
Client Review          6.527
dtype: float64
```

I used the following methods to handle them:

'Dependent Count', 'Prior Claims' : fill with zero

'Years Lived', 'Yearly Earnings', 'Wellness Index', 'Automobile Age', 'Financial Rating', 'Coverage Period' : fill with median

'Relationship Status', 'Client Review': fill with mode

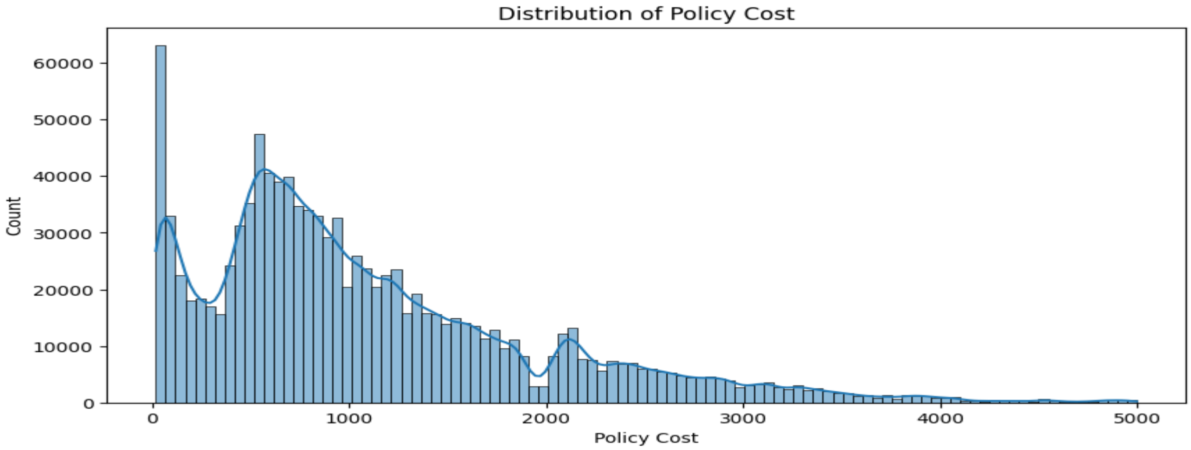
Fill Job Title missing with 'Unknown'

2.3 feature engineering

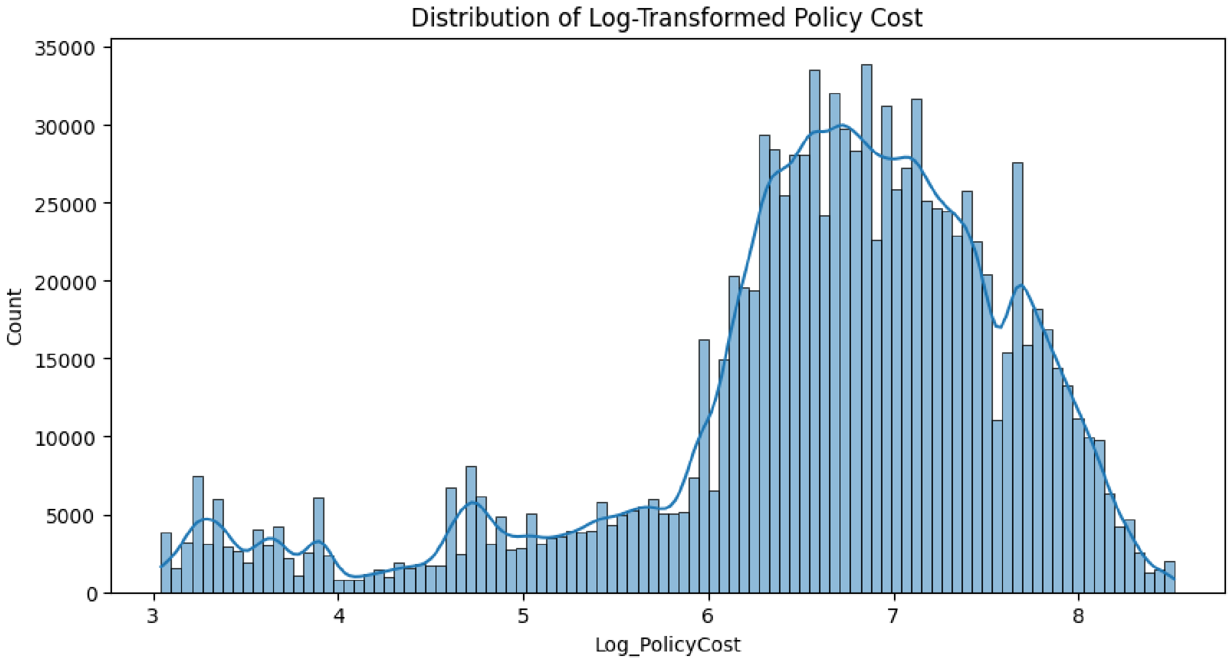
In the feature engineering phase, temporal features were extracted from the 'Coverage Commencement' column, which was initially a date string. This column was first converted into a proper datetime format, and then various time-based attributes were derived, including the year, month, day, day of the week, and day of the year. Additionally, a seasonal feature was created by mapping the coverage month to one of the four seasons—Spring, Summer, Autumn, or Winter—based on standard month groupings. These new temporal features aimed to capture potentially meaningful patterns related to time and seasonality in the policy data. After extracting the relevant information, the original 'Coverage Commencement' column was dropped from both the training and test datasets to avoid redundancy.

2.4 visualization

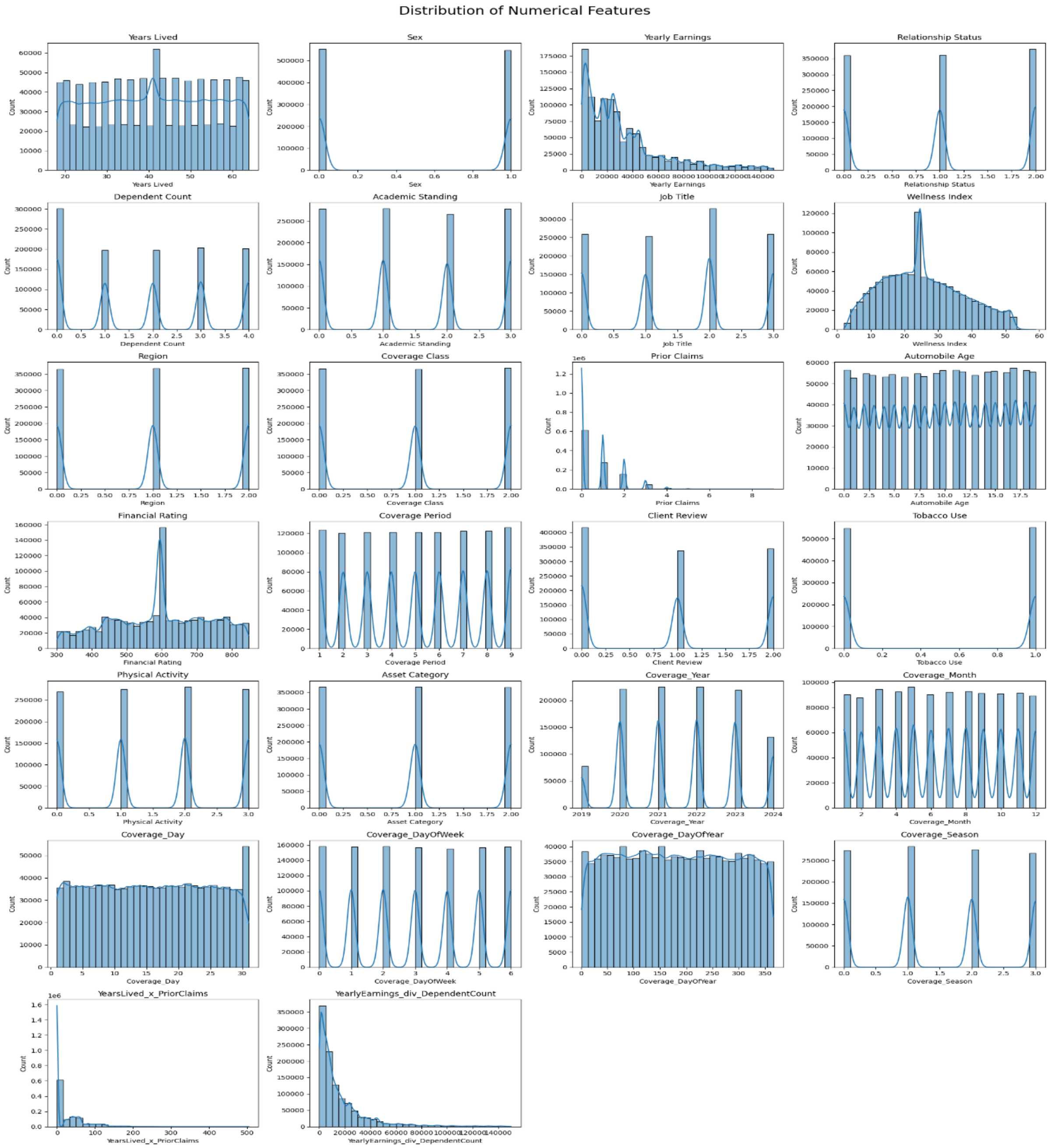
In this section we will go through some visualization related to the data that can help us to have a better sense of our data and the relationship between the features.



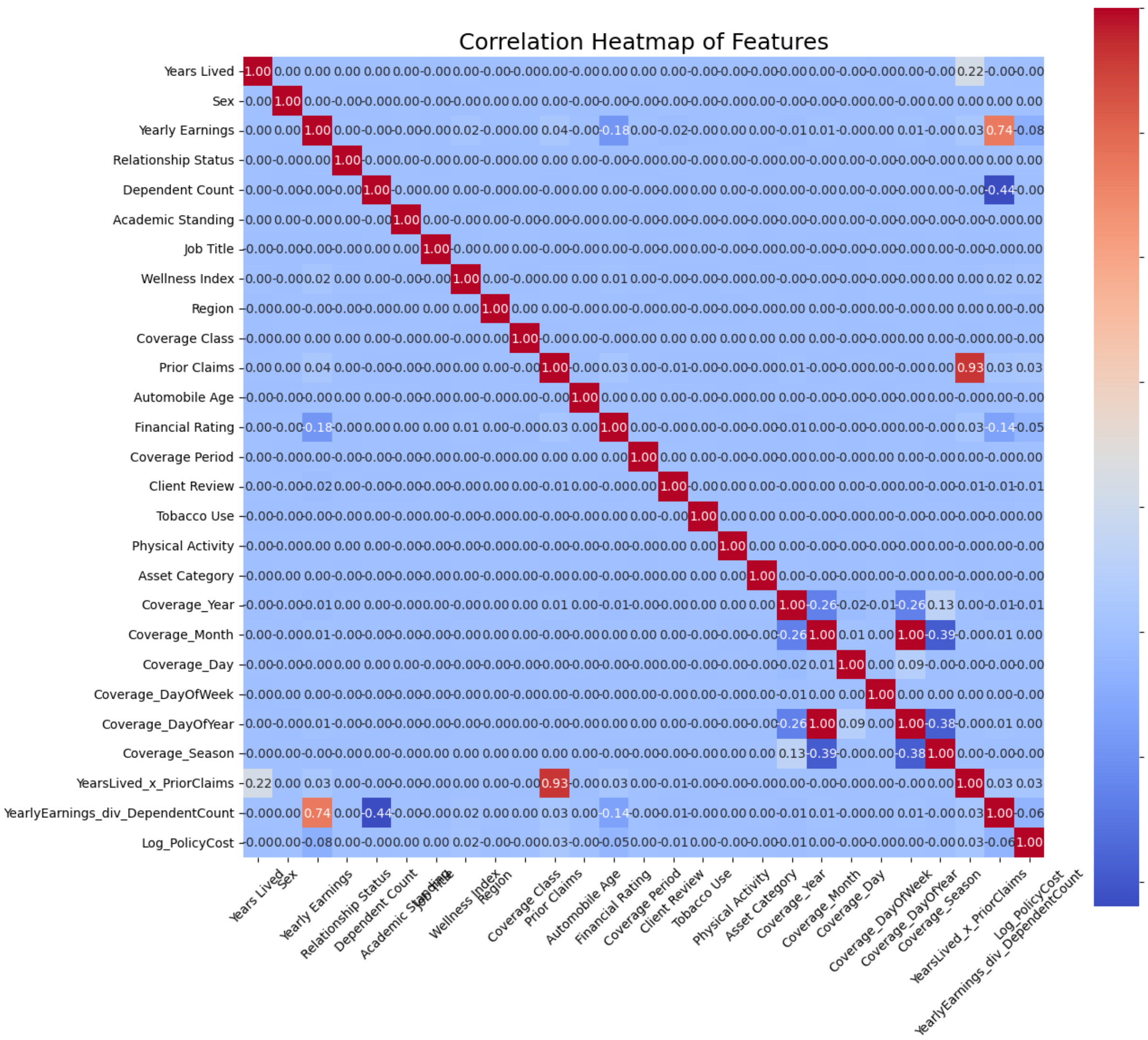
The histogram titled "Distribution of Policy Cost" illustrates the frequency of policy costs across a range from $0 to $5000, with bins segmented at $1000 intervals. The data shows a right-skewed distribution, indicating that the majority of policies are concentrated at the lower end of the cost spectrum, particularly between $0 and $1000, where the count peaks at approximately 60,000. As the policy cost increases, the frequency declines sharply, with fewer policies exceeding $2000 and a minimal number reaching the upper limit of $5000. This pattern suggests that high-cost policies are relatively rare compared to their low-cost counterparts, which dominate the dataset. The visualization effectively highlights the prevalence of affordable policies and the diminishing occurrence of more expensive ones.



The histogram titled "Distribution of Log-Transformed Policy Cost" displays the frequency of policy costs after applying a logarithmic transformation, with values ranging from approximately 3 to 8 on the log scale. The distribution appears roughly bell-shaped, suggesting a more symmetric and normalized pattern compared to the original right-skewed data.



Here we can see the distribution of all the features.



The correlation heatmap reveals that most features in the dataset have weak or no linear correlation with each other, indicating low multicollinearity. The strongest positive correlation (0.93) is between the engineered feature YearsLived_x_PriorClaims and Prior Claims, which is expected due to direct mathematical dependence. Similarly, YearlyEarnings_div_DependentCount has a strong correlation with Yearly Earnings (0.74) and a moderate negative correlation with Dependent Count (-0.44), again due to its constructed nature. Aside from these, most correlations are close to zero, suggesting largely independent features, which can be beneficial for many machine learning models.

3 Model 1: Linear Regression (with scaling)

A Linear Regression model was trained on the scaled features using StandardScaler to normalize the input data, ensuring that each feature contributed equally to the model. The model was fitted on the training set and evaluated on the validation set using the RMSLE (Root Mean Squared Logarithmic Error) metric to account for the skewed distribution of the target variable. Predictions were made on the validation set, and the logarithmic outputs were converted back to their original scale using the inverse transformation np.expm1. The final model was then applied to the test set to generate predictions, which were formatted into the submission template and saved as a CSV file named 'submission_linear_regression.csv', which was also made available for download via Google Colab.

Linear Regression RMSLE: 1.0900

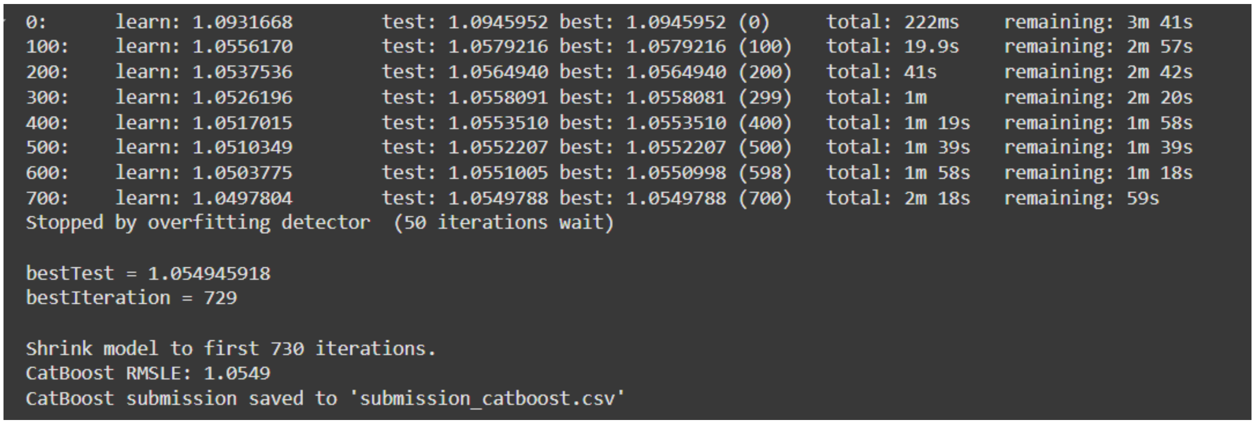
4 Model 2: Random Forest Regressor

A Random Forest Regressor was trained as the second model using 100 decision trees and a maximum depth of 15 to capture non-linear relationships and interactions in the data without requiring feature scaling. The model was fitted on the original training features and evaluated on the validation set using the RMSLE metric, which is suitable for skewed targets such as policy cost. After validating the model's performance, predictions were generated for the test set, and the logarithmic outputs were inverse-transformed to return to the original scale. These predictions were inserted into the submission template and saved as 'submission_random_forest.csv', which was then made available for download via Google Colab.

Random Forest RMSLE: 1.0544

5 Model 3: GBM (CatBoost)

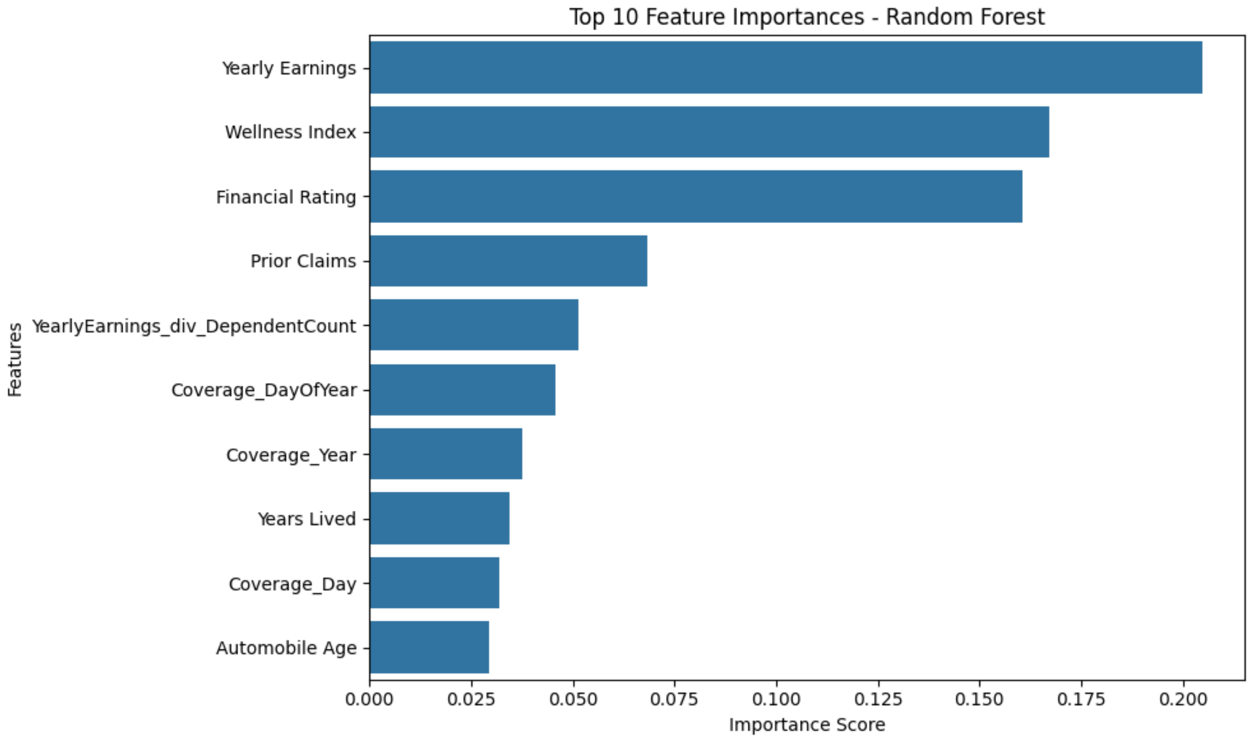
The third model applied was a CatBoost Regressor, a gradient boosting algorithm optimized for handling categorical features and complex data structures. The model was configured with 1000 iterations, a learning rate of 0.05, a maximum tree depth of 6, and used RMSE as the loss function, with early stopping based on the validation set to prevent overfitting. After training on the original (unscaled) training data, the model's performance was evaluated using the RMSLE metric on the validation set to measure predictive accuracy on a logarithmic scale. Predictions were then generated for the test set, inverse-transformed from the log scale using np.expm1, and inserted into the sample submission format. The final predictions were saved as 'submission_catboost.csv' and made available for download via Google Colab.

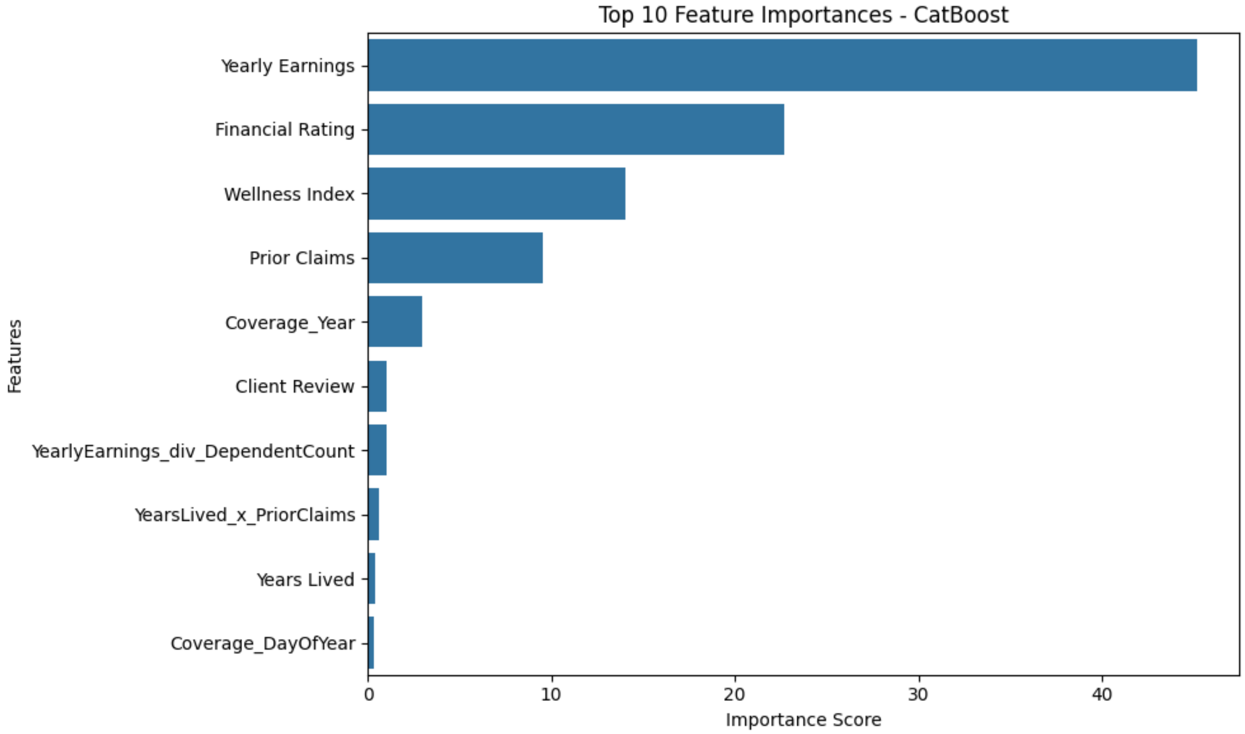


```
0:      learn: 1.0931668        test: 1.0945952 best: 1.0945952 (0)     total: 222ms    remaining: 3m 41s
100:    learn: 1.0556170        test: 1.0579216 best: 1.0579216 (100)   total: 19.9s    remaining: 2m 57s
200:    learn: 1.0537536        test: 1.0564940 best: 1.0564940 (200)   total: 41s      remaining: 2m 42s
300:    learn: 1.0526196        test: 1.0558091 best: 1.0558081 (299)   total: 1m       remaining: 2m 20s
400:    learn: 1.0517015        test: 1.0553510 best: 1.0553510 (400)   total: 1m 19s   remaining: 1m 58s
500:    learn: 1.0510349        test: 1.0552207 best: 1.0552207 (500)   total: 1m 39s   remaining: 1m 39s
600:    learn: 1.0503775        test: 1.0551005 best: 1.0550998 (598)   total: 1m 58s   remaining: 1m 18s
700:    learn: 1.0497804        test: 1.0549788 best: 1.0549788 (700)   total: 2m 18s   remaining: 59s
Stopped by overfitting detector  (50 iterations wait)

bestTest = 1.054945918
bestIteration = 729

Shrink model to first 730 iterations.
CatBoost RMSLE: 1.0549
CatBoost submission saved to 'submission_catboost.csv'
```

6 Model Interpretation

Numbers are not enough. Therefore, we reserved a slice of our analysis for feature importance using impurity-based measures from both the Random Forest and CatBoost models. The findings reveal that features related to **financial rating, wellness index, and yearly earnings** are the strongest drivers for policy cost. In particular, *Yearly Earnings* consistently ranked as the most important feature across both models, followed closely by *Financial Rating* and *Wellness Index*. Additional influential features included *Prior Claims*, *Coverage Year*, and *Years Lived*, along with engineered variables like *YearlyEarnings divided by Dependent Count* and *Coverage Day of Year*. These results suggest that policy cost is heavily influenced by the customer's financial status, health indicators, and historical claim behavior. Two concrete suggestions based on these findings are: (1) enhance the accuracy and completeness of financial and wellness data collection processes, as these variables are crucial for model performance; and (2) incorporate these top features into risk assessment and pricing strategies to create more tailored, data-driven insurance products that balance competitiveness with profitability.


Top 10 Feature Importances - Random Forest
Features
Yearly Earnings
Wellness Index
Financial Rating
Prior Claims
YearlyEarnings_div_DependentCount
Coverage_DayOfYear
Coverage_Year
Years Lived
Coverage_Day
Automobile Age
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Importance Score


Top 10 Feature Importances - CatBoost
Features
Yearly Earnings
Financial Rating
Wellness Index
Prior Claims
Coverage_Year
Client Review
YearlyEarnings_div_DependentCount
YearsLived_x_PriorClaims
Years Lived
Coverage_DayOfYear
0
10
20
30
40
Importance Score

Conclusion

In this project, we developed a robust regression pipeline to predict **Policy Cost** using a complex, anonymized insurance dataset. By methodically preprocessing the data—handling missing values, engineering meaningful temporal and interaction features, and applying effective visualization and correlation analysis—we built a strong foundation for model development. We implemented and compared three different models: **Linear Regression**, **Random Forest Regressor**, and **CatBoost Regressor**. Among them, the **CatBoost Regressor** achieved the best performance based on the **Root Mean Squared Logarithmic Error (RMSLE)** metric, capturing intricate non-linear relationships and handling categorical data efficiently. Our feature importance analysis revealed that **Yearly Earnings**, **Financial Rating**, and **Wellness Index** are the most influential drivers of policy cost. This insight provides practical value: it suggests the company should prioritize the collection and validation of financial and wellness-related data, and leverage these features more directly in pricing strategies. In summary, this analysis highlights how thoughtful data preparation, diverse modeling approaches, and interpretability techniques can lead to actionable insights and predictive accuracy. These methods can help insurance companies design fairer pricing models, identify high-risk profiles more effectively, and ultimately drive data-informed business decisions.

References

https://www.kaggle.com/competitions/data-science-5-sbu/overview

Maryam Soleimani 401222075