Best Machine Learning Algorithms for Predicting Motor Insurance Claims

OUR TEAM

Festus Godwin

(Team Lead)

Rofhiwa Ntshagovhe

(Admin Lead)

Peter Maila

Shamsuddeen Lawal

Sandisiwe Mtsha

Kasavuli Mark

OUTLINE

01 Project Overview

02 Analytical Insights

03 Data Processing

04 Models Comparative Analysis

05 Recommendations

06 Conclusion

PROJECT OVERVIEW

SIGNIFICANCE

- This project holds significant importance in optimizing insurance operations

PROBLEM

- Suboptimal Predictions
- Fraudulent Claims

SOLUTION

- Optimize Claims Prediction
- Promote Stakeholders’ Trust

ANALYTICAL INSIGHTS

DATASETS

DATASET A

Number of Observations: 161 832

Number of Features: 23

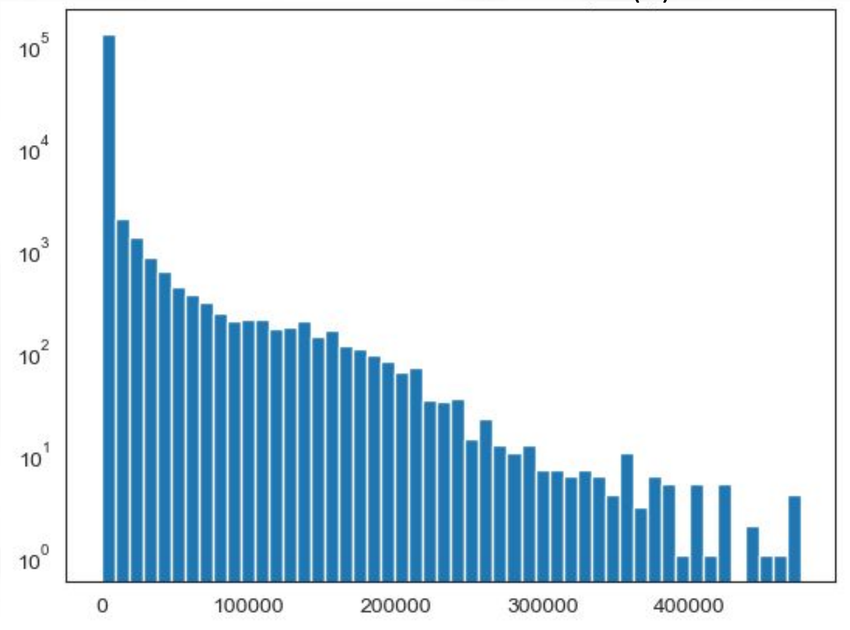
DATASET B

Number of Observations: 25 519

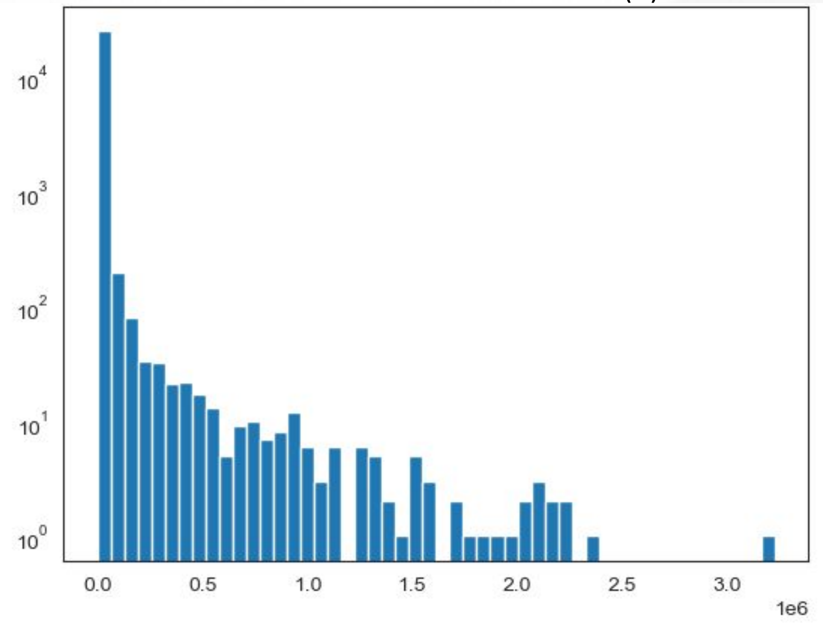
Number of Features: 9

DATA DISTRIBUTION

Distribution of Annual Claims(A)



Distribution of Annual Claims(B)



EXPLORATORY DATA ANALYSIS

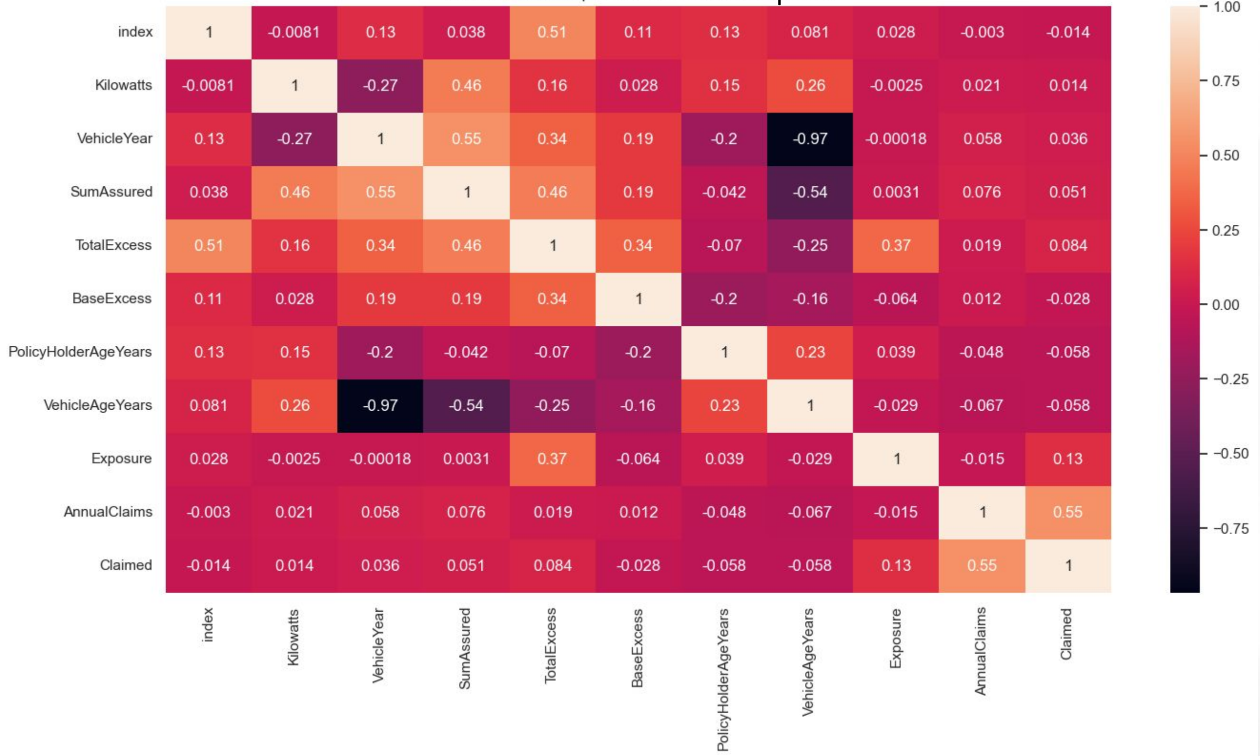
	count	mean	std	min	25%	50%	75%	max
Kilowatts	129465.0	84.202819	31.655294	0.0	63.000000	74.0	97.0	426.0
VehicleYear	129465.0	2014.476414	3.653251	2004.0	2012.000000	2015.0	2017.0	2022.0
SumAssured	129465.0	157700.526397	72162.021462	30000.0	109100.000000	146700.0	189300.0	515000.0
TotalExcess	129465.0	24369.043502	12218.025355	0.0	19000.000000	25130.0	31770.0	90460.0
BaseExcess	129465.0	4480.361812	1980.984924	0.0	4000.000000	5000.0	5000.0	60000.0
PolicyHolderAgeYears	129465.0	38.158792	10.118095	18.0	30.000000	36.0	44.0	92.0
VehicleAgeYears	129465.0	5.599622	3.635835	0.0	3.000000	5.0	8.0	16.0
Exposure	129465.0	0.755729	0.319860	0.1	0.421918	1.0	1.0	1.0
AnnualClaims	129465.0	4309.076033	22902.575946	0.0	0.000000	0.0	0.0	475900.0

EXPLORATORY DATA ANALYSIS

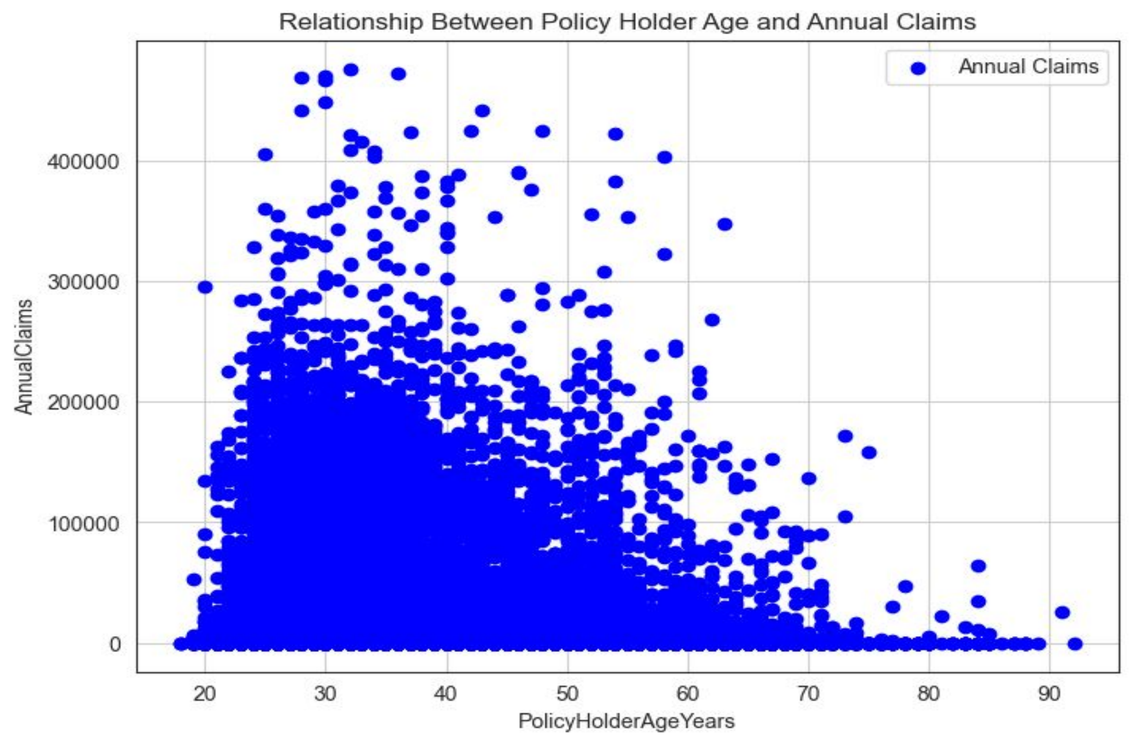
	vehicle_year	vehicle_age	sum_insured	excess	exposure	annual_claims
count	25519.000000	25519.000000	2.551900e+04	25519.000000	25519.000000	2.551900e+04
mean	2016.004154	6.065285	6.650507e+05	66036.699497	0.505176	8.691689e+03
std	5.964126	5.891118	6.849623e+05	68156.828771	0.329561	8.643850e+04
min	1972.000000	0.000000	2.000000e+03	2500.000000	0.083333	0.000000e+00
25%	2014.000000	2.000000	2.100000e+05	20691.000000	0.250000	0.000000e+00
50%	2018.000000	5.000000	3.406880e+05	33206.300000	0.333333	0.000000e+00
75%	2020.000000	9.000000	9.000240e+05	89276.000000	0.916666	0.000000e+00
max	2023.000000	51.000000	1.000000e+07	1000000.000000	1.000000	3.226904e+06

EXPLORATORY DATA ANALYSIS

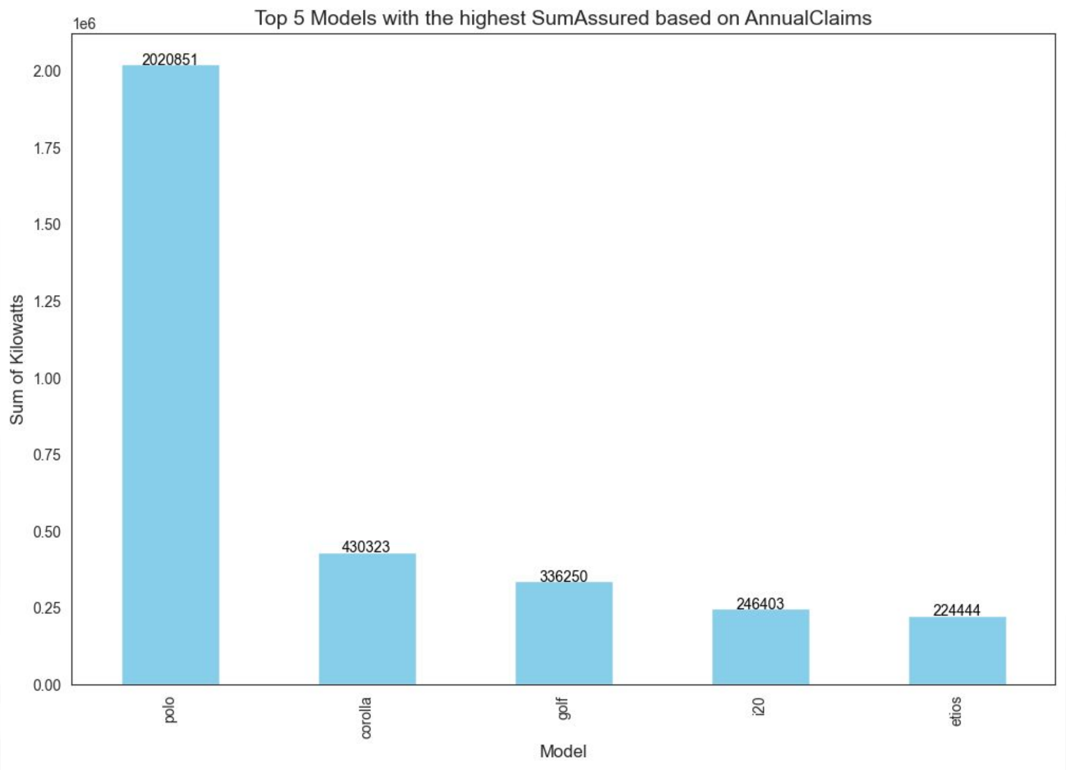
Correlation Heatmap



EXPLORATORY DATA ANALYSIS



EXPLORATORY DATA ANALYSIS



DATA PROCESSING

PRE PROCESSING

Missing/Null Values

Missing Values Imputed Using the Mode:

- Area
- Occupation

FEATURE ENGINEERING

Encoding Categorical Data

- Binary encoding
- Target encoding
- Label encoding
- CatBoost encoding

FEATURE ENGINEERING

Feature Selection

Features Removed:

- Area
- Occupation
- Make
- Colour

ML ALGORITHMS

General Linear Model

Random Forest

CatBoost, XGBoost, LightGBM, Explainable Boosting Machines (EBM)

MODELS COMPARATIVE ANALYSIS

MODEL IMPLEMENTATION

```
smote = SMOTE(sampling_strategy='auto', random_state=42, k_neighbors=10)
X_resampled_classification, y_resampled_classification = smote.fit_resample(X_classification, y_classification)

# Split the resampled data into train and test sets for classification
X_train_classification, X_test_classification, y_train_classification, y_test_classification = train_test_split(X_resampled_classification,
                                                                                                                        y_resampled_classification, test_size=0.2,
                                                                                                                        random_state=42)

# Train the binary classification model (Random Forest)
from sklearn.ensemble import RandomForestClassifier

classification_model = RandomForestClassifier(class_weight={0: 0.5, 1: 0.5}, random_state=42)
classification_model.fit(X_train_classification, y_train_classification)

classification_predictions = classification_model.predict(X_test_classification)

# Classification report for Random Forest Classifier with custom class names
target_names = ['No Claims', 'Claimed']
classification_report_result = classification_report(y_test_classification, classification_predictions, target_names=target_names)
print("Classification Report:\n", classification_report_result)
```

```
Classification Report:
               precision    recall  f1-score   support

   No Claims       0.89      0.95      0.92     23061
     Claimed       0.94      0.88      0.91     23349

    accuracy                           0.91     46410
   macro avg       0.92      0.91      0.91     46410
weighted avg       0.92      0.91      0.91     46410
```

MODEL IMPLEMENTATION

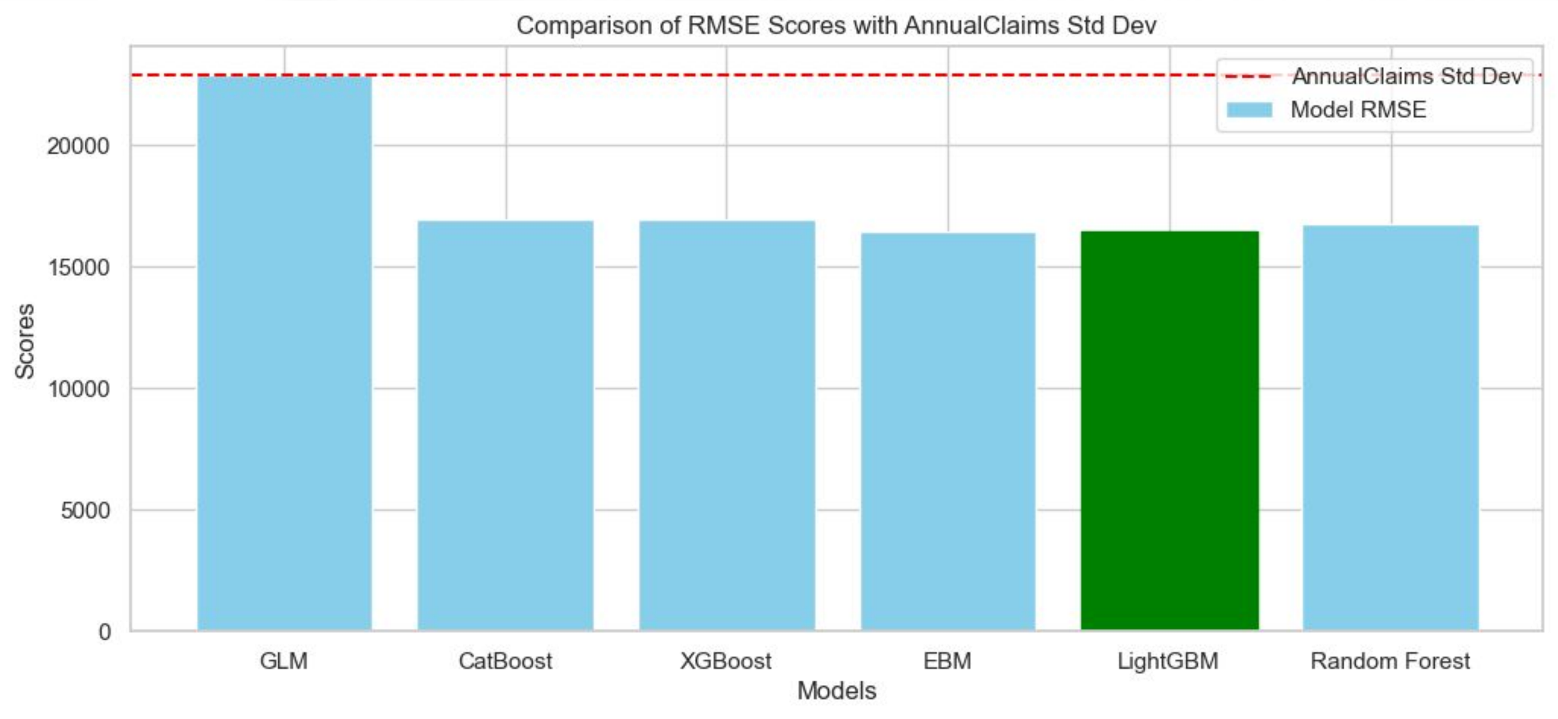
LightGBM Regressor

```
# LightGBM
lightgbm_model = LGBMRegressor(objective='tweedie', tweedie_variance_power=1.6, metric='rmse', verbose=-1)
lightgbm_model.fit(X_regression, y_regression, sample_weight=train_data['Exposure'])

#Make Predictions
lightgbm_predictions = lightgbm_model.predict(test_data.drop(['AnnualClaims'], axis=1))

#Calculate RMSE
lightgbm_rmse = mean_squared_error(test_data['AnnualClaims'], lightgbm_predictions, squared=False)
print("LightGBM RMSE:", lightgbm_rmse)
```

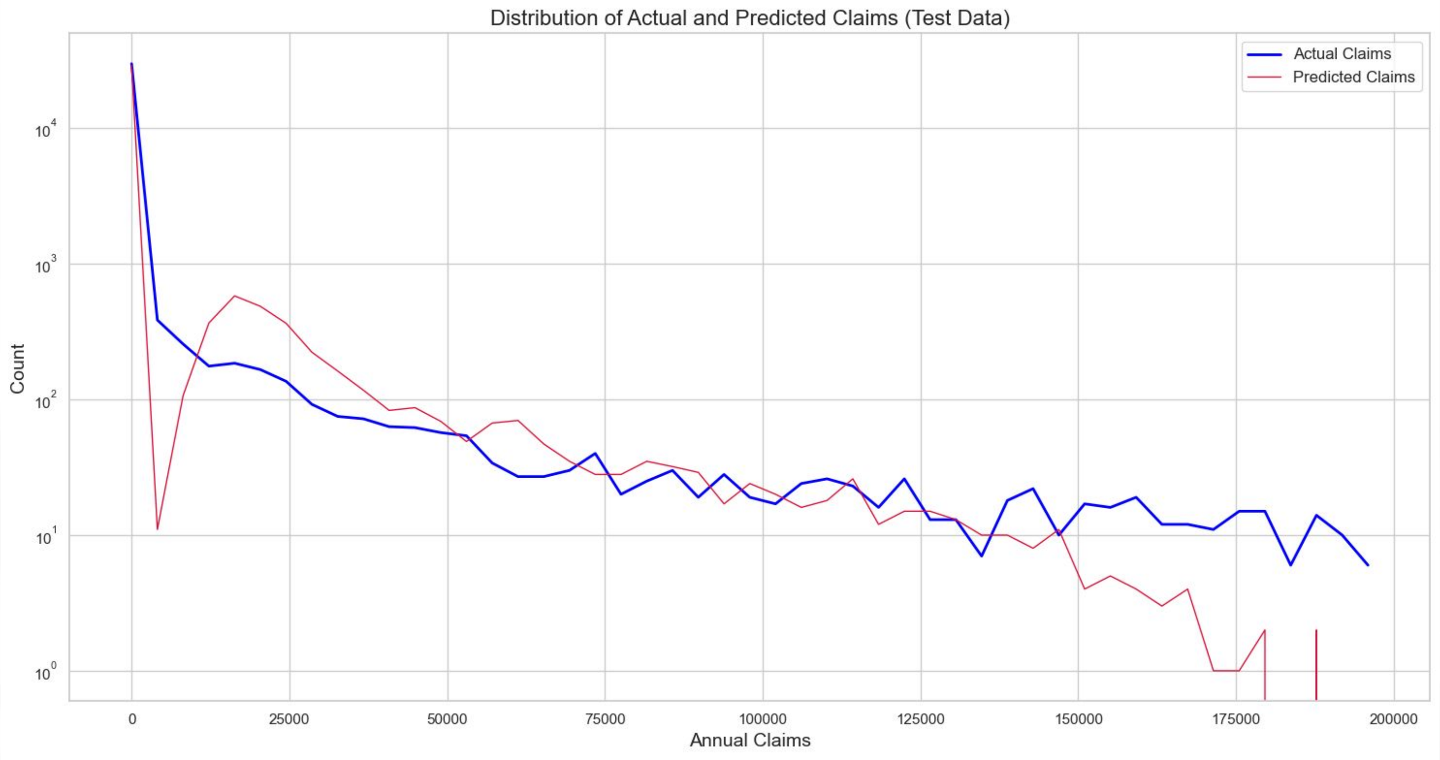
✓ 0.9s

```
LightGBM RMSE: 16451.559711261558
```

COMPARATIVE ANALYSIS



COMPARATIVE ANALYSIS

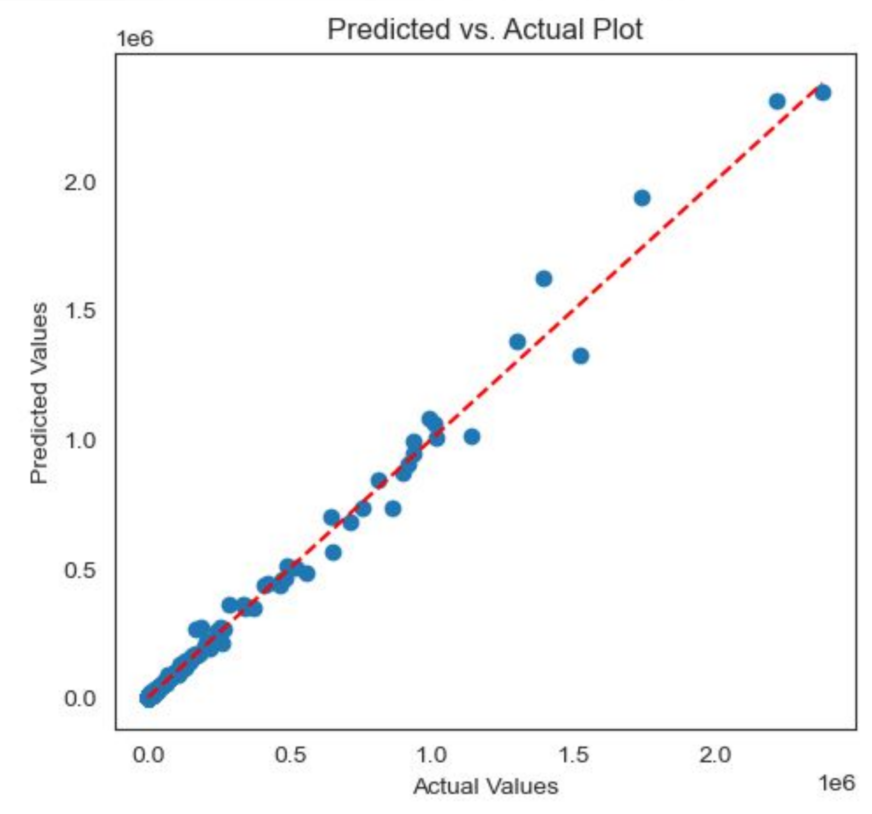


MODEL IMPLEMENTATION

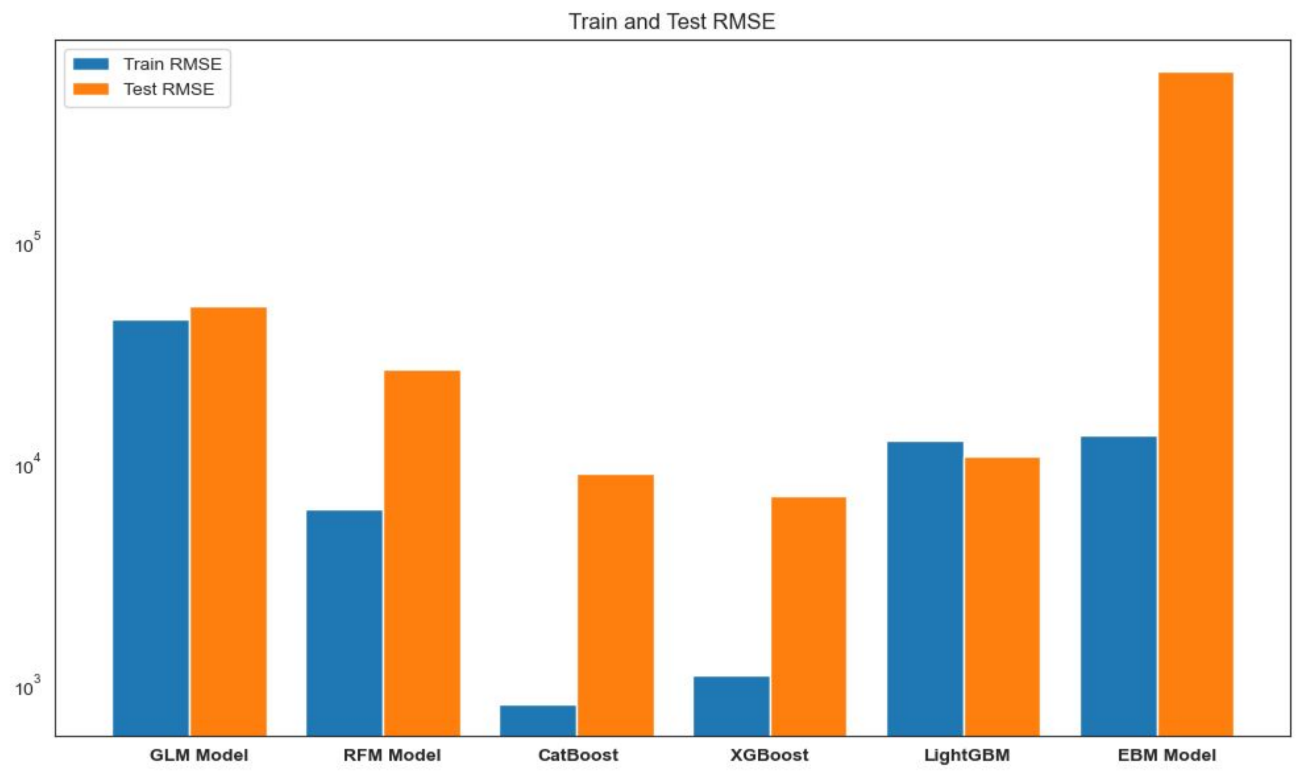
```python
# Evaluation
print(f'MAE Evaluation scores for training and validation')
xgb_train_mae = round(mae(y_train_cat, xgb_train_y_pred),2)
print(f'Train XGBoost MAE: {xgb_train_mae}')
xgb_test_mae = round(mae(y_test_cat, xgb_test_y_pred),2)
print(f'Test XGBoost MAE: {xgb_test_mae}')

print('-' * 50)

print(f'RMSE Evaluation scores for training and validation')
xgb_train_rmse = round(np.sqrt(mse(y_train_cat, xgb_train_y_pred)),2)
print(f'Train XGBoost RMSE: {xgb_train_rmse}')
xgb_test_rmse = round(np.sqrt(mse(y_test_cat, xgb_test_y_pred)),2)
print(f'Test XGBoost RMSE: {xgb_test_rmse}')
```

```
--------------------------------------------------
MAE Evaluation scores for training and validation
Train XGBoost MAE: 202.28
Test XGBoost MAE: 641.83
--------------------------------------------------
RMSE Evaluation scores for training and validation
Train XGBoost RMSE: 1112.62
Test XGBoost RMSE: 7151.81
```

MODEL IMPLEMENTATION



COMPARATIVE ANALYSIS



RECOMMENDATIONS

01

Best ML Algorithm

- XGBoost
- LightGBM

02 Continuous Algorithm Assessment

03 Education and Awareness

CONCLUSION

THANK YOU

CONNECT WITH US ON LINKEDIN

Festus Godwin

Rofhiwa Ntshagovhe

Peter Maila

Shamsuddeen Lawal

Sandisiwe Mtsha

Mark Kasavuli