Car Insurance Claim Prediction
Data Science Project

Diego Alonso Meza Delgado

THE PROBLEM

Insurance companies face significant challenges due to the financial risks associated with policy claims. In the context of car insurance, accurately predicting which policies will result in claims can provide substantial benefits. By leveraging car, policy, and demographic features, we aim to develop a predictive model that helps mitigate financial risks and improve decision-making processes.

THE DATA

Kaggle dataset with 44 columns and 58,592 rows. Each column represent a feature, including the target variable. Each row represent a policy record.

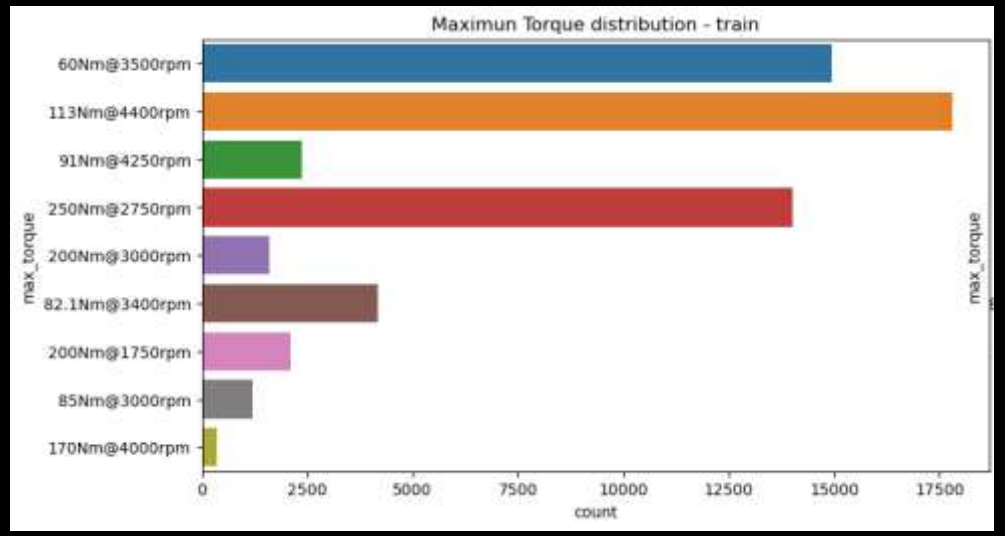
	policy_id	policy_tenure	age_of_car	age_of_policyholder	area_cluster	population_density	make	segment	model	fuel_type	...
0	ID00001	0.515874	0.05	0.644231	C1	4990	1	A	M1	CNG	...
1	ID00002	0.672619	0.02	0.375000	C2	27003	1	A	M1	CNG	...
2	ID00003	0.841110	0.02	0.384615	C3	4076	1	A	M1	CNG	...
3	ID00004	0.900277	0.11	0.432692	C4	21622	1	C1	M2	Petrol	...
4	ID00005	0.596403	0.11	0.634615	C5	34738	2	A	M3	Petrol	...

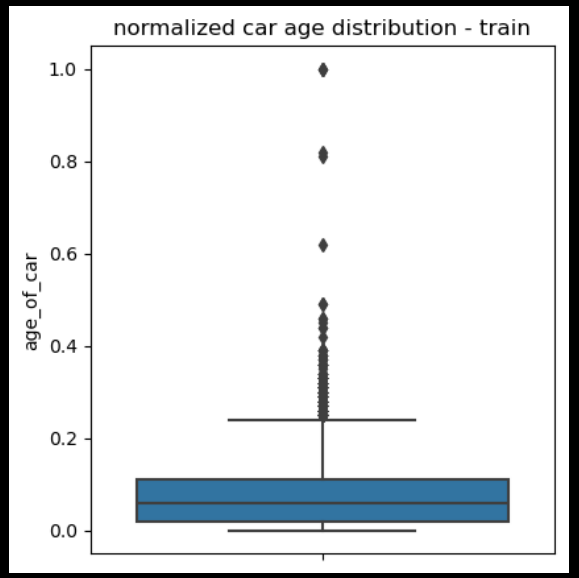
DATA WRANGLING

- Assess the data types
- Encode categories
- Split columns
- Check for missing values
- Verify ID Uniqueness
- Check for duplicates

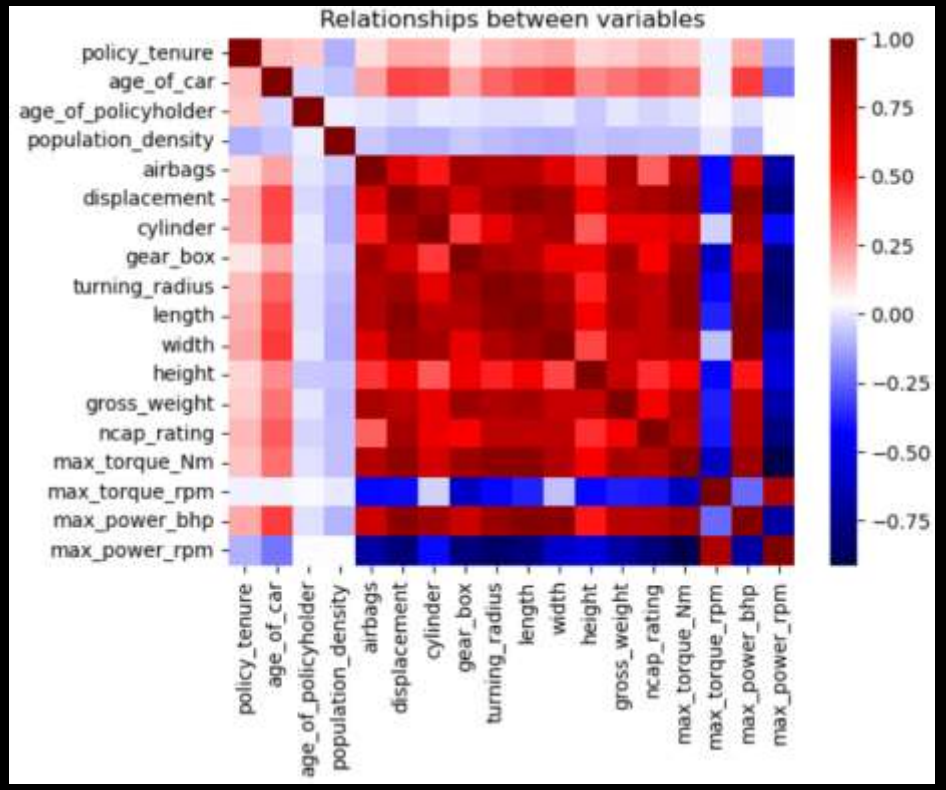
EXPLORATORY DATA ANALYSIS (EDA)

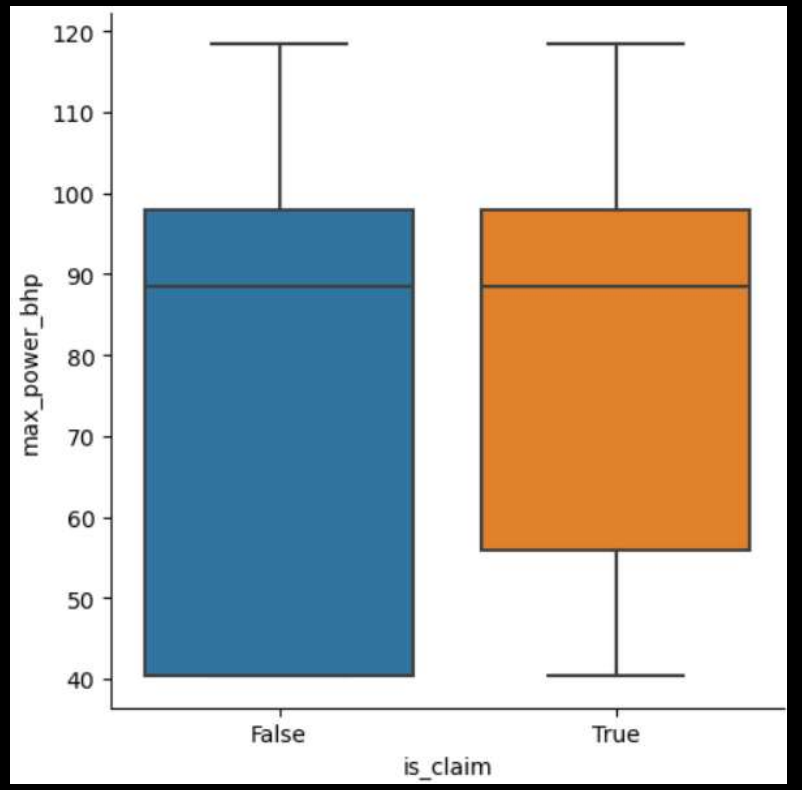
- Explore each variable separately
- Explore relations between variables
- Explore relations between the features and the target variable





EXPLORATORY DATA ANALYSIS (EDA)





EXPLORATORY DATA ANALYSIS (EDA)

- Proportions difference null hypothesis significance tests
- Means difference null hypothesis significance tests

```
segment  is_claim
A        False       16275
         True         1046
B1       False        3929
         True          244
B2       False       17058
         True         1256
C1       False        3329
         True          228
C2       False       13117
         True          901
Utility  False        1136
         True           73
Name: count, dtype: int64
```

```
stat, pval = proportions_ztest([244, 1256], [3929 + 244, 17058 + 1256])

if pval < 0.05:
    print("Reject the null hypothesis: The proportions are different")
else:
    print("Fail to reject the null hypothesis: There is not enough evidence to affirm that the proportions are different")
print(pval)
```

```
Reject the null hypothesis: The proportions are different
0.018164796314432102
```

EXPLORATORY DATA ANALYSIS (EDA)

```
sample1 = train[train["is_claim"] == True].policy_tenure
sample2 = train[train["is_claim"] == False].policy_tenure

# Variance Homogeneity Test
stat_levene, pval_levene = levene(sample1, sample2)

if pval_levene < 0.05: # If Variance Homogeneity use t test to compare distributions
    print("Variance Homogeneity Assumption")
    stat, pval = ttest_ind(sample1, sample2)
    if pval < 0.05:
        print("Reject the null hypothesis: The distributions means are different")
    else:
        print("Fail to reject the null hypothesis: There is not enough evidence to affirm that the distributions means ar
    print(pval)
else: # Use permutations to compare distributions
    print("No Variance Homogeneity")
    perm_distributions_comparison(sample1, sample2)
```

```
Variance Homogeneity Assumption
Reject the null hypothesis: The distributions means are different
3.0181155800813767e-81
```

PREPROCESSING

- Get dummies for categorical variables
- Scale the numeric variables
- Split the data in training and test sets

```
train_dummies = pd.get_dummies(train.drop(columns=["policy_id"]))
```

```
scaler = StandardScaler()
```

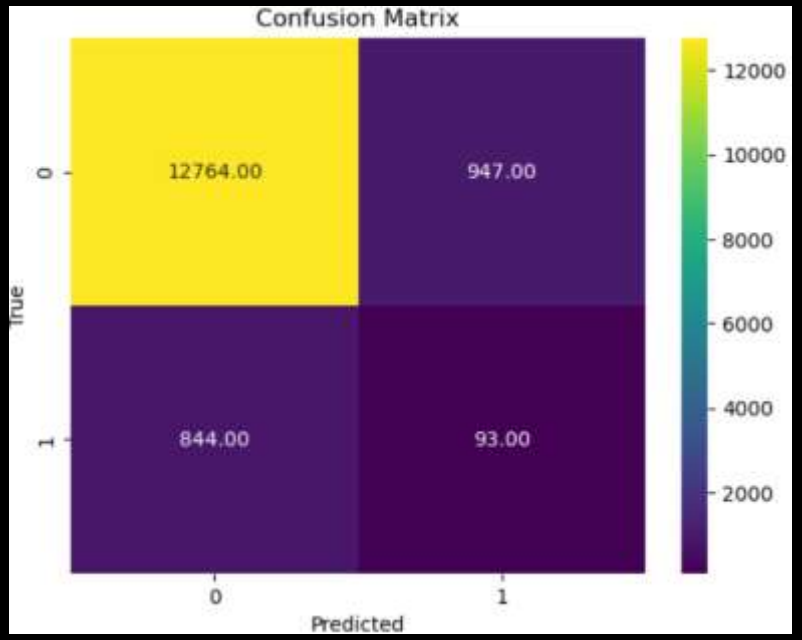
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,stratify=y, random_state=24)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

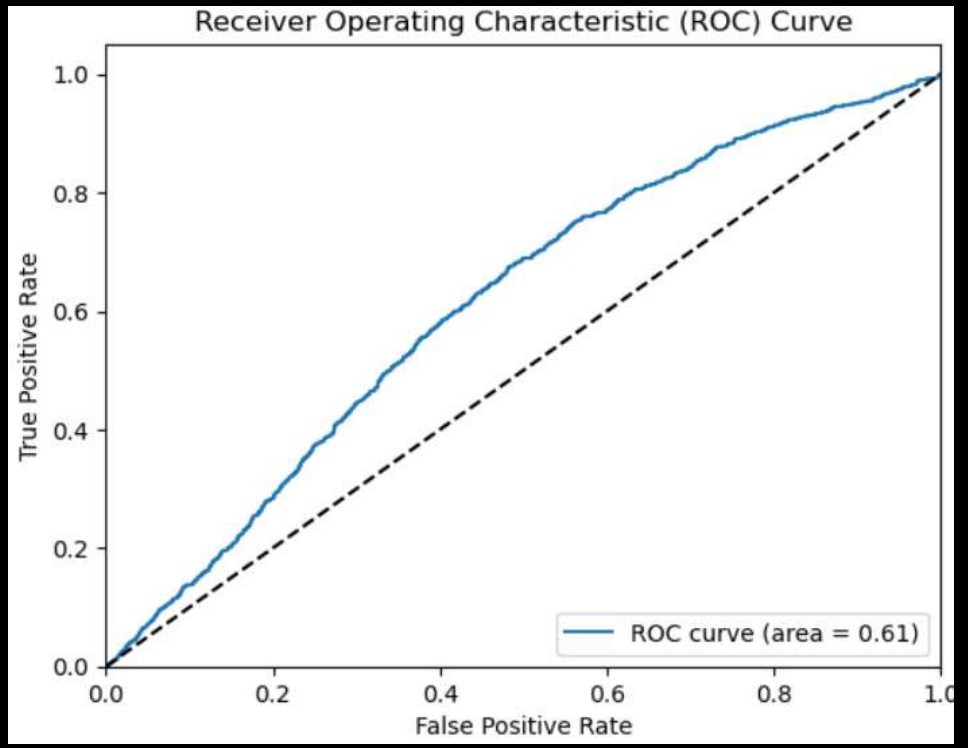
MODELLING, EVALUATION AND OPTIMIZATION

- Baseline Models: K-Nearest Neighbors Classifier, Decision Tree Classifier, and Logistic Regression
- Ensemble Methods: Random Forest Classifier and Extreme Gradient Boosting
- Hyperparameter Tuning: Random Search
- SMOTE: Synthetic Minority Over-sampling Technique
- Advanced Model: Multilayer Perceptron Classifier (MPC)
- Evaluation Metrics: Classification Report (precision, recall, f1 score), Confusion Matrix, Receiver Operating Characteristic (ROC) curve and its Area Under the Curve (AUC)

RANDOM FOREST CLASSIFIER (RANDOM SEARCH)

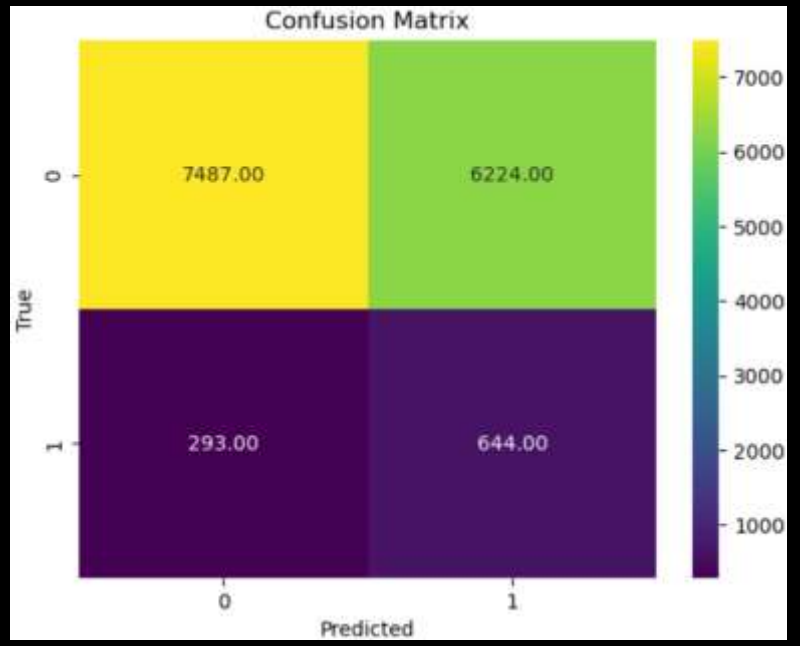
	precision	recall	f1-score	support
False	0.94	0.93	0.93	13711
True	0.09	0.10	0.09	937
accuracy			0.88	14648
macro avg	0.51	0.52	0.51	14648
weighted avg	0.88	0.88	0.88	14648

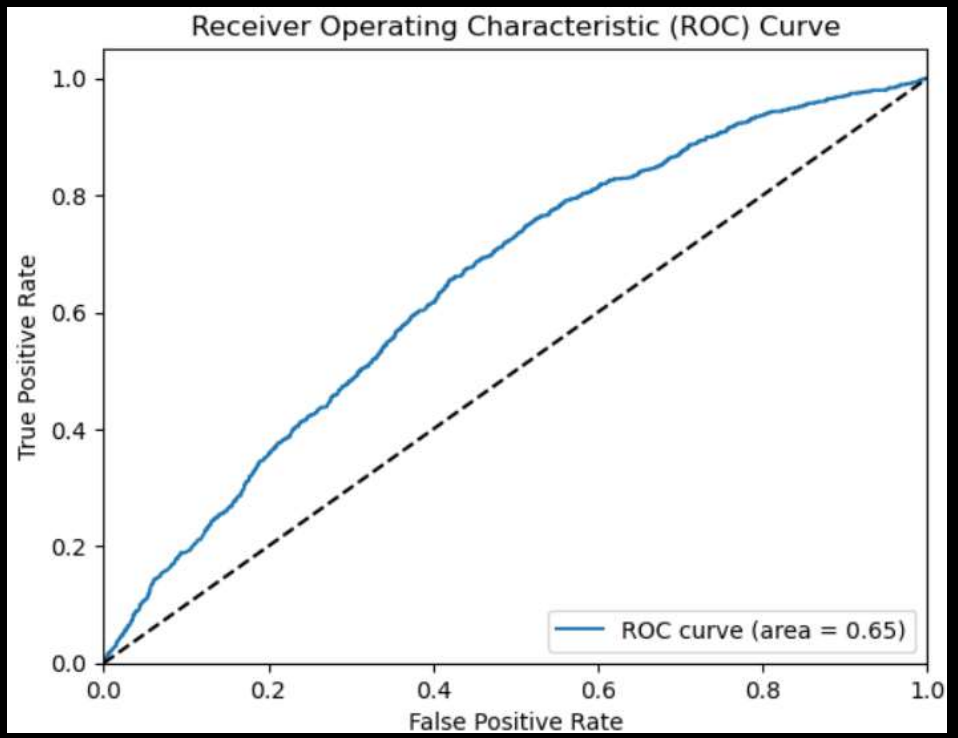




EXTREME GRADIENT BOOSTING (RANDOM SEARCH)

	precision	recall	f1-score	support
False	0.96	0.55	0.70	13711
True	0.09	0.69	0.17	937
accuracy			0.56	14648
macro avg	0.53	0.62	0.43	14648
weighted avg	0.91	0.56	0.66	14648





BUSINESS MODEL USING COST FUNCTION

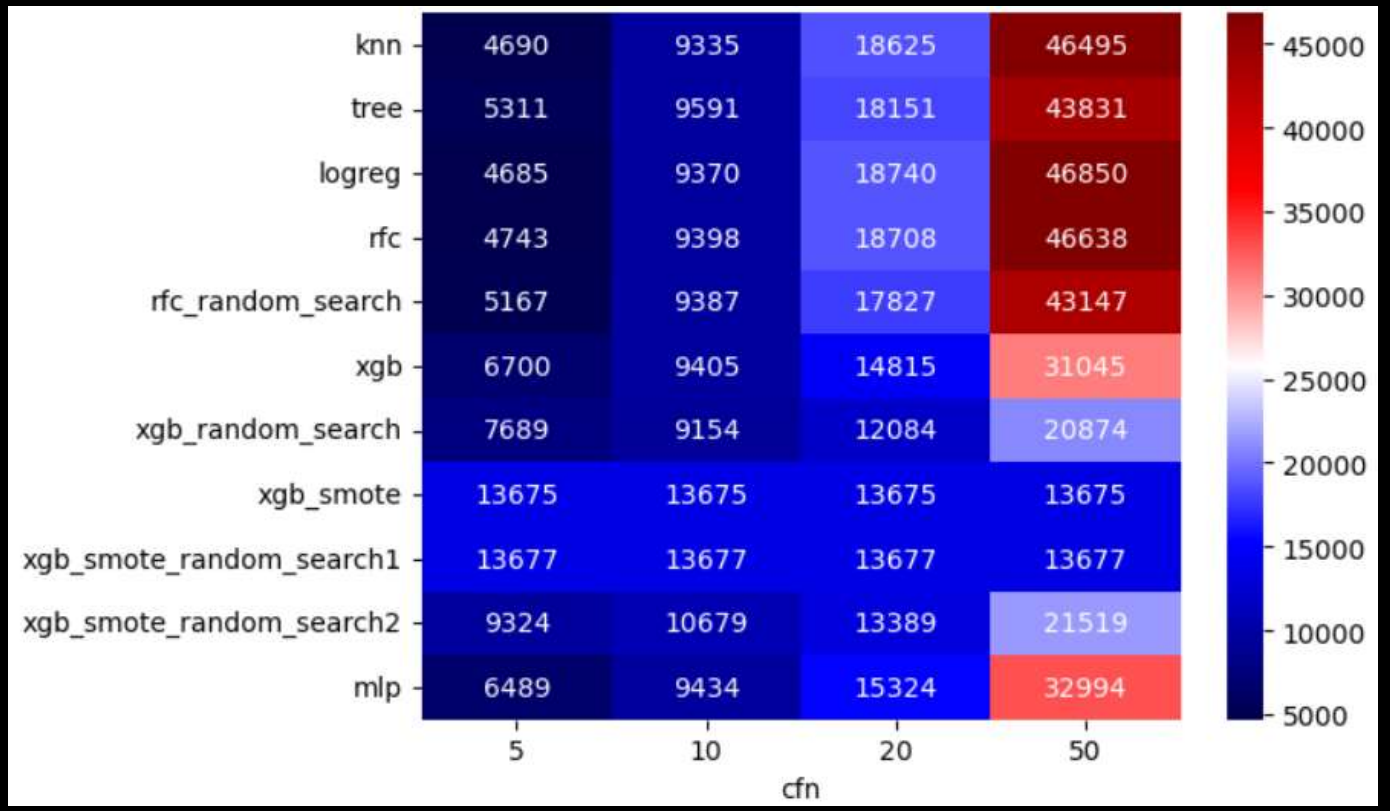
```
base_knn = {"fp": 45, "fn": 929}
base_tree = {"fp": 1031, "fn": 856}
base_logreg = {"fp": 0, "fn": 937}

base_rfc = {"fp": 88, "fn": 931}
rfc_random_search = {"fp": 947, "fn": 844}

base_xgb = {"fp": 3995, "fn": 541}
xgb_random_search = {"fp": 6224, "fn": 293}

xgb_smote = {"fp": 13675, "fn": 0}
xgb_smote_random_search1 = {"fp": 13677, "fn": 0}
xgb_smote_random_search2 = {"fp": 7969, "fn": 271}

mlp = {"fp": 3544, "fn": 589}
```

$$C = (CFP * FP) + (CFN * FN)$$

- CFP: The cost of a false positive
- CFN: The cost of a false negative
- FP: The number of false positives
- FN: The number of false negatives

BUSINESS MODEL USING COST FUNCTION

```
cfp = 1
cfn_list = [5, 10, 20, 50]
```



CONCLUSION

- Best Model: Extreme Gradient Boosting with Random Search
- Best Hyperparameters: {'n_estimators': 200, 'max_depth': 6, 'learning_rate': 0.01, 'gamma': 0}
- Metrics: {AUC: 0.65, accuracy: 0.56, , f1_score (True) = 0.17, business_cost_10 = 9154, business_cost_20 = 12084}
- Final Recommendations: Improve the dataset's class imbalance. Increase computational resources to perform a Complete Grid Search for hyperparameter tuning. Establish a precise ratio between false positives and false negatives

THANKS