Customer Claim Prediction *at* Singlife Travel Insurance

Presented by: Sulthan Mahdi M. D.

Table of contents

01

Business Problem

SingLife Travel Insurance, based in **Singapore**, is a **digital life insurance company** renowned for its innovative financial products. Specializing in the insurance sector, the company offers a range of digital insurance products, with a particular emphasis on **travel insurance**. Their coverage caters to both leisure and business travelers, reflecting their commitment to providing comprehensive and convenient insurance solutions.

Problem and Goals

Despite SingLife's commitment to delivering insurance services, it has **encountered challenges** in efficiently **handling customer insurance claims.** To address this issue, the company is **seeking the expertise of a data scientist.** The data scientist's role would involve **analyzing** and **predicting** whether a customer is likely to file an insurance claim. This predictive analysis **aims to enhance the company's claim handling processes**, ensuring more effective and timely responses to customer needs.

Create a Machine Learning model that can be used by insurance companies to predict which customers will claim/not claim. Machine learning Models must be able to minimize losses to the smallest possible extent.

Metrics Evaluation

- **FN**: The company estimates a loss of 800 SGD for cases where customers who actually make a claim (default) go undetected. This figure encompasses the loss from both the enrolled customers who file claims and the marketing costs incurred in acquiring new customers.

- **FP:** Meanwhile, the loss for situations where customers do not make a claim but are erroneously identified as claimants is 4000 SGD. This amount represents the average insurance coverage cost for these customers.

We need to use fscore metrics with beta 1/5, this is because the precision value is 5x more important than the recall value

Data UnderStanding

02

Data

1. Agency	Name of agency
2. Agency Type	Type of travel insurance agencies
3. Distribution Channel	Channel of travel insurance agencies
4. Product Name	Name of the travel insurance products
5. Gender	Gender of insured
6. Duration	Duration of travel
7. Destination	Destination of travel
8. Net Sales	Amount of sales of travel insurance policies
9. Commission (in value)	Commission received for travel insurance
10. Age	Age of insured
11. Claim	Claim status

Data

	Agency	Agency Type	Distribution Channel	Product Name	Gender	Duration	Destination	Net Sales	Commision (in value)	Age	Claim
0	C2B	Airlines	Online	Annual Silver Plan	F	365	SINGAPORE	216.0	54.0	57	No
1	EPX	Travel Agency	Online	Cancellation Plan	NaN	4	MALAYSIA	10.0	0.0	33	No
2	JZI	Airlines	Online	Basic Plan	M	19	INDIA	22.0	7.7	26	No
3	EPX	Travel Agency	Online	2 way Comprehensive Plan	NaN	20	UNITED STATES	112.0	0.0	59	No
4	C2B	Airlines	Online	Bronze Plan	M	8	SINGAPORE	16.0	4.0	28	No

```
RangeIndex: 44328 entries, 0 to 44327
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Agency                44328 non-null  object
 1   Agency Type           44328 non-null  object
 2   Distribution Channel  44328 non-null  object
 3   Product Name          44328 non-null  object
 4   Gender                12681 non-null  object
 5   Duration              44328 non-null  int64
 6   Destination           44328 non-null  object
 7   Net Sales             44328 non-null  float64
 8   Commision (in value)  44328 non-null  float64
 9   Age                   44328 non-null  int64
 10  Claim                 44328 non-null  object
dtypes: float64(2), int64(2), object(7)
```

```
df.isna().sum()/len(df)*100
```

✓ 0.0s

```
Agency                   0.000000
Agency Type              0.000000
Distribution Channel     0.000000
Product Name             0.000000
Gender                  71.392799
Duration                 0.000000
Destination              0.000000
Net Sales                0.000000
Commision (in value)     0.000000
Age                      0.000000
Claim                    0.000000
dtype: float64
```

```
df["Claim"].value_counts()
```

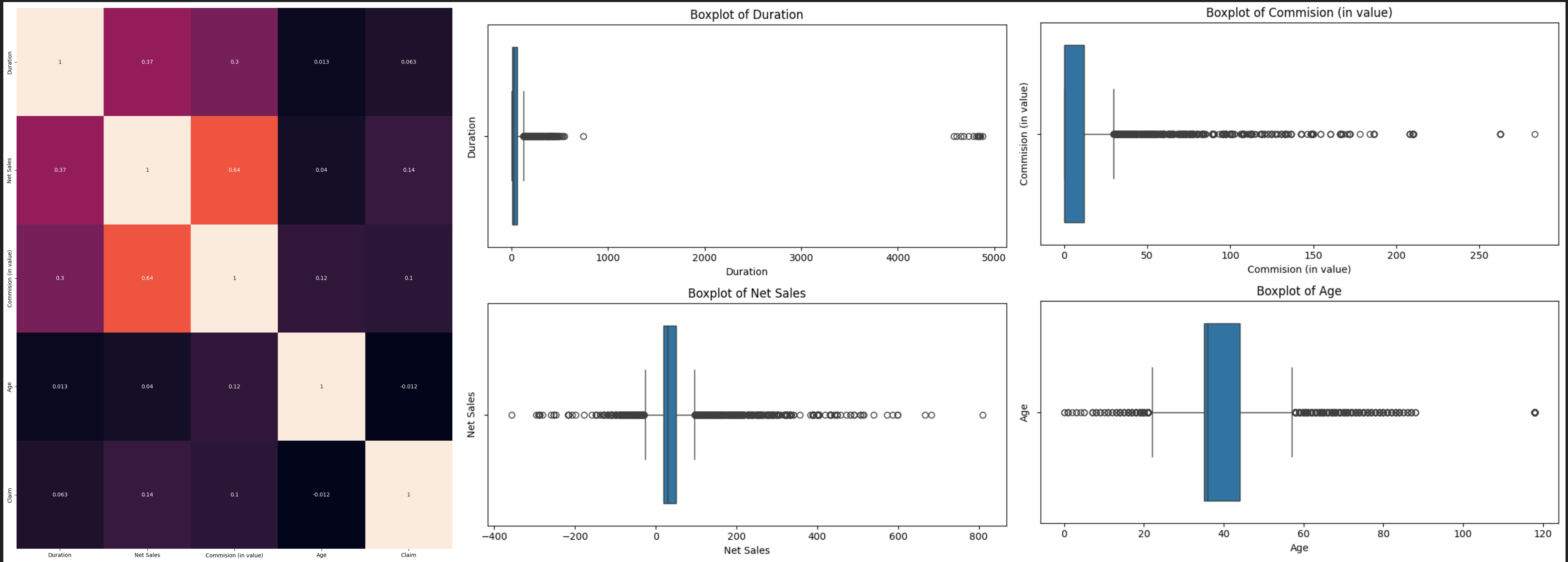
✓ 0.0s

```
Claim
No     38651
Yes      673
Name: count, dtype: int64
```

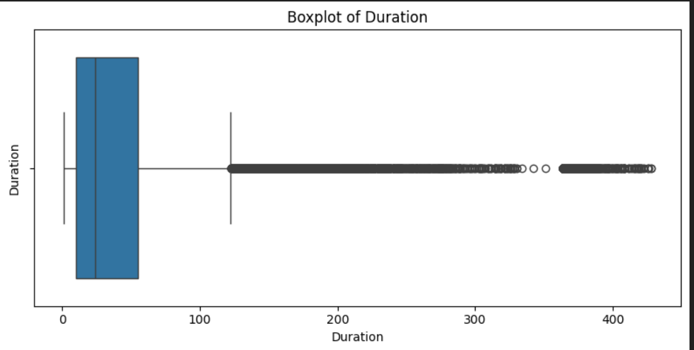
03

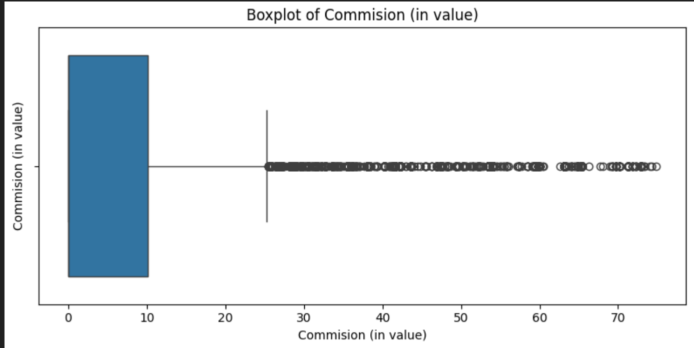
Data Cleaning & Preprocessing

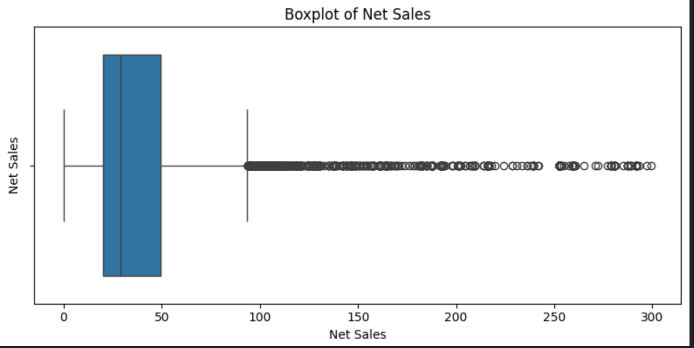
Numerical Before Cleaning

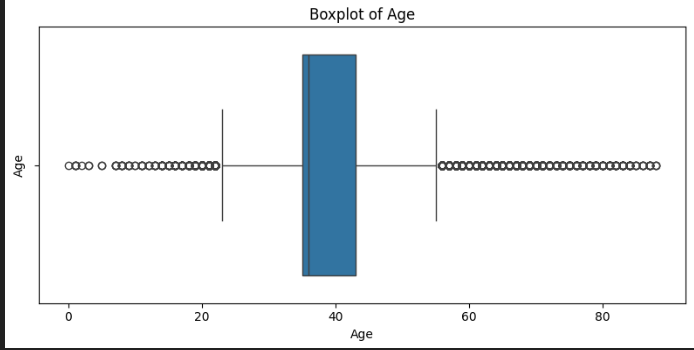


Numerical After Cleaning

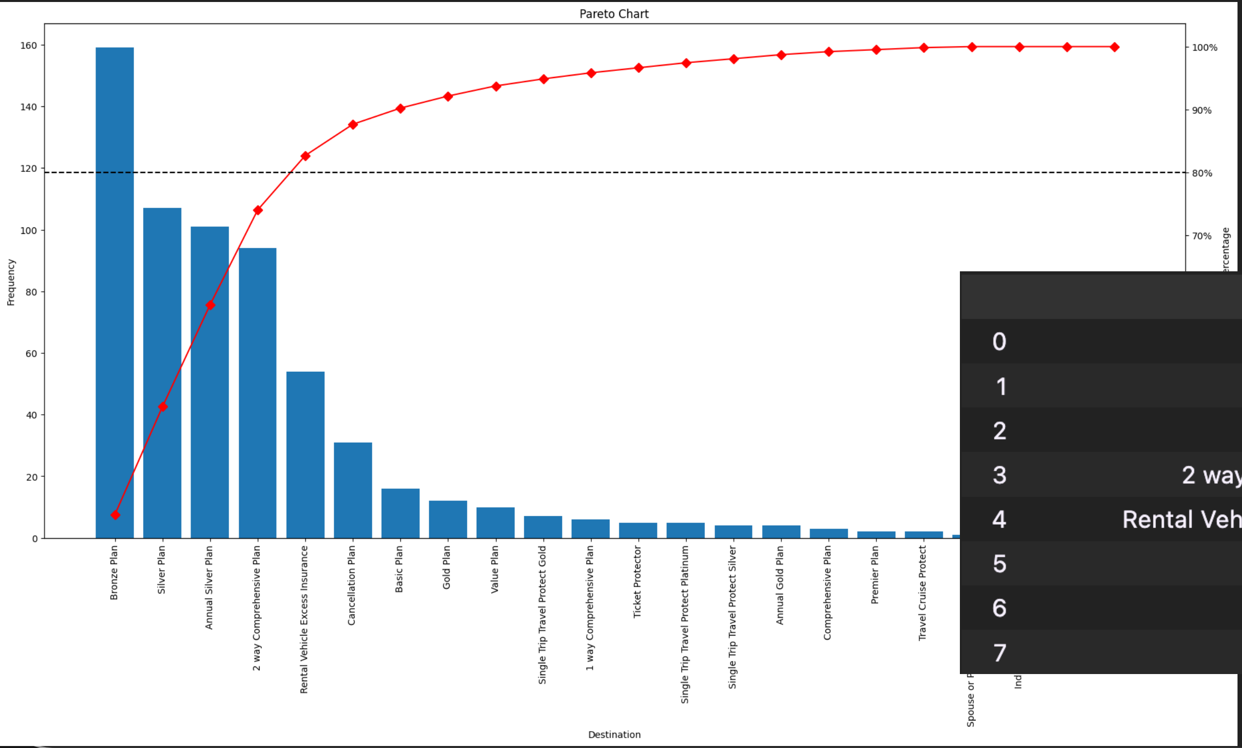






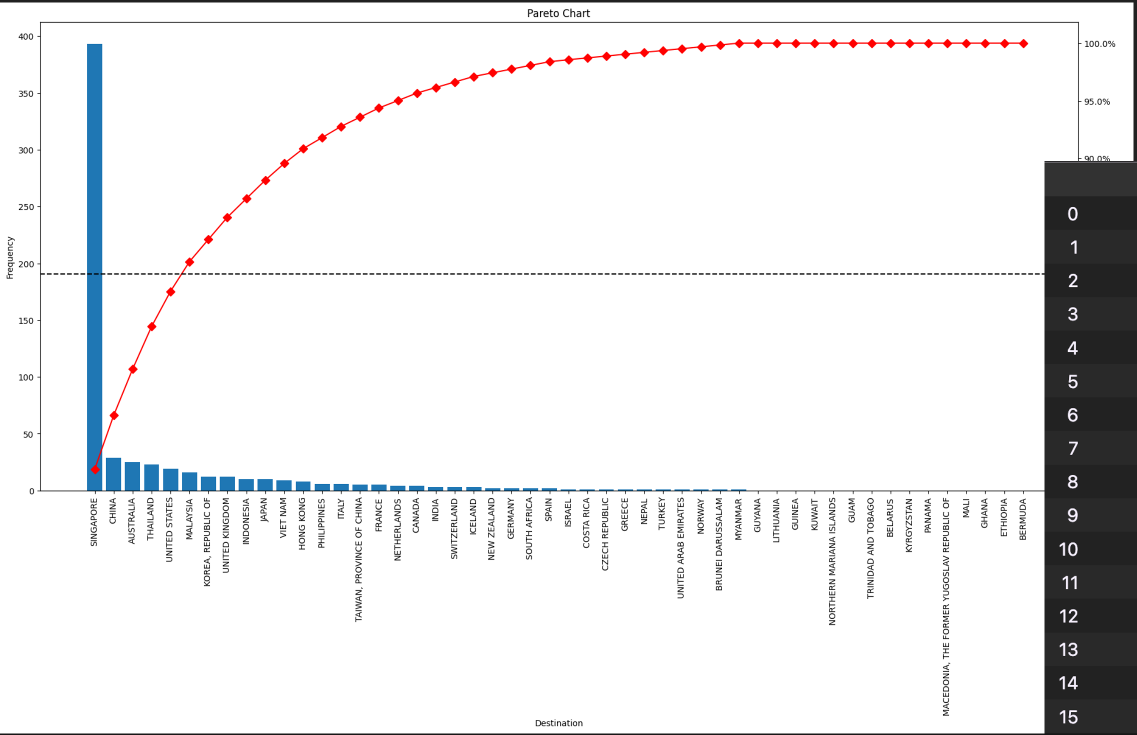


Categorical - Product Name



	Product_Name	Total_Claim	Cum_Percentage
0	Bronze Plan	159	25.52
1	Silver Plan	107	42.70
2	Annual Silver Plan	101	58.91
3	2 way Comprehensive Plan	94	74.00
4	Rental Vehicle Excess Insurance	54	82.66
5	Cancellation Plan	31	87.64
6	Basic Plan	16	90.21
7	Gold Plan	12	92.13

Categorical - Destination

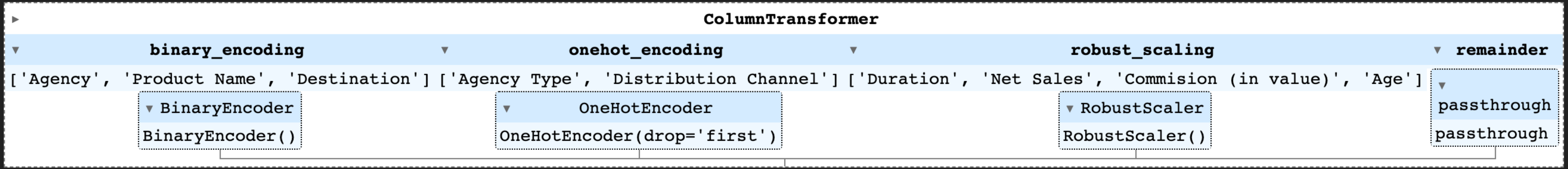


	Destination_Name	Total_Claim	Cum_Percentage
0	SINGAPORE	393	63.08
1	CHINA	29	67.74
2	AUSTRALIA	25	71.75
3	THAILAND	23	75.44
4	UNITED STATES	19	78.49
5	MALAYSIA	16	81.06
6	KOREA, REPUBLIC OF	12	82.99
7	UNITED KINGDOM	12	84.91
8	INDONESIA	10	86.52
9	JAPAN	10	88.12
10	VIET NAM	9	89.57
11	HONG KONG	8	90.85
12	PHILIPPINES	6	91.81
13	ITALY	6	92.78
14	TAIWAN, PROVINCE OF CHINA	5	93.58
15	FRANCE	5	94.38

Column Transformer

```python
transformer = ColumnTransformer([
    ("binary_encoding", BinaryEncoder(), ["Agency", "Product Name", "Destination"]),
    ("onehot_encoding", OneHotEncoder(drop="first"), ["Agency Type", "Distribution Channel"]),
    ("robust_scaling", RobustScaler(), ['Duration', 'Net Sales', 'Commision (in value)', 'Age'])
], remainder= "passthrough")
transformer
```

✓ 0.0s Pythoı



Analytics (Modeling)

04

Cross Validation

```python
f1per5_scorer = make_scorer(fbeta_score, beta=1/5)
```

```python
# Cross Validation
models = [logreg, knn, tree, voting, stacking, bagging, rf, adaboost, gboost]

for i in models:
    pipe_model = Pipeline([
        ("preprocessing", transformer),
        ("modeling",i)
    ])

    model_cv = cross_val_score(
        estimator= pipe_model,
        X = X_train,
        y = y_train,
        cv = 5,
        scoring = f1per5_scorer,
        error_score = "raise",
        n_jobs = -1
    )
```

Cross Validation Score

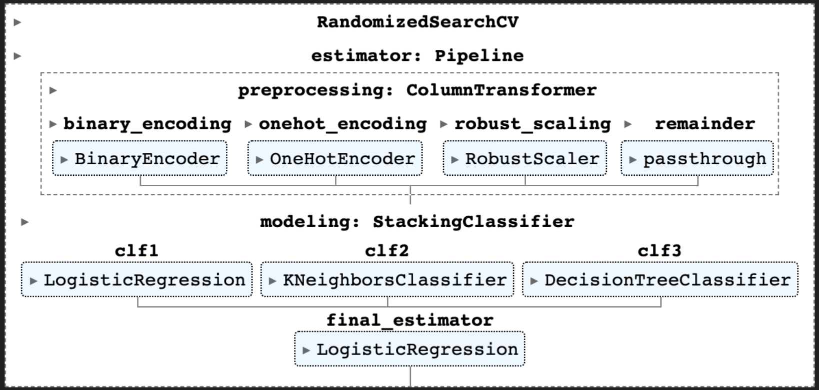
	algo	all_score	mean_score	std_score
0	LogisticRegression(random_state=0)	[0.7303, 0.8016, 0.6648, 0.7973, 0.7758]	0.753945	0.051268
4	StackingClassifier(estimators=[('clf1', Logist...	[0.7383, 0.8068, 0.6557, 0.7937, 0.7671]	0.752311	0.053740
3	VotingClassifier(estimators=[('clf1', Logistic...	[0.6935, 0.8016, 0.651, 0.7875, 0.7535]	0.737410	0.057083
8	GradientBoostingClassifier(random_state=0)	[0.6603, 0.7785, 0.6437, 0.7818, 0.7405]	0.720947	0.058388
7	AdaBoostClassifier(random_state=0)	[0.6758, 0.7531, 0.6478, 0.7403, 0.7241]	0.708248	0.039992
5	BaggingClassifier(estimator=KNeighborsClassifi...	[0.6765, 0.7698, 0.6305, 0.7362, 0.7255]	0.707670	0.048826
6	RandomForestClassifier(random_state=0)	[0.6683, 0.7863, 0.6384, 0.7403, 0.7049]	0.707639	0.052156
1	KNeighborsClassifier()	[0.6727, 0.7758, 0.6178, 0.7438, 0.7181]	0.705655	0.055358
2	DecisionTreeClassifier(random_state=0)	[0.6054, 0.6864, 0.6185, 0.6347, 0.6453]	0.638058	0.027728

Hyperparameter Tuning

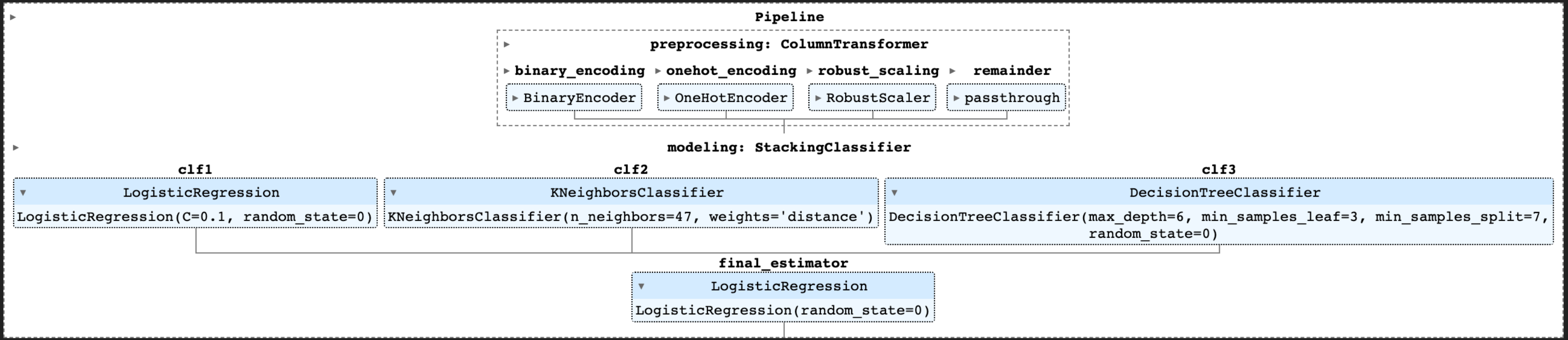
```python
# Hyperparameter tuning
hyperparam = {
    "modeling__clf1__C": np.logspace(5, -5, 11),
    "modeling__clf1__solver": ['lbfgs', 'liblinear'],
    "modeling__clf2__n_neighbors": range(1, 50, 2),
    "modeling__clf2__weights": ['uniform', 'distance'],
    "modeling__clf3__criterion": ['gini', 'entropy', 'log_loss'],
    "modeling__clf3__max_depth": range(2, 50, 1),
    "modeling__clf3__min_samples_split": range(2, 50),
    "modeling__clf3__min_samples_leaf": range(1, 25),
    "modeling__final_estimator": [logreg, knn, tree]
}

pipe_model = Pipeline([
    ("preprocessing", transformer),
    ("modeling", stacking)
])

randomsearch = RandomizedSearchCV(
    estimator= pipe_model,
    cv = 5,
    n_jobs= -1,
    scoring= f1per5_scorer,
    param_distributions= hyperparam,
    random_state=0,
    n_iter=10000
)
randomsearch
```



Hyperparameter Tuning Best Estimator



Best Threshold

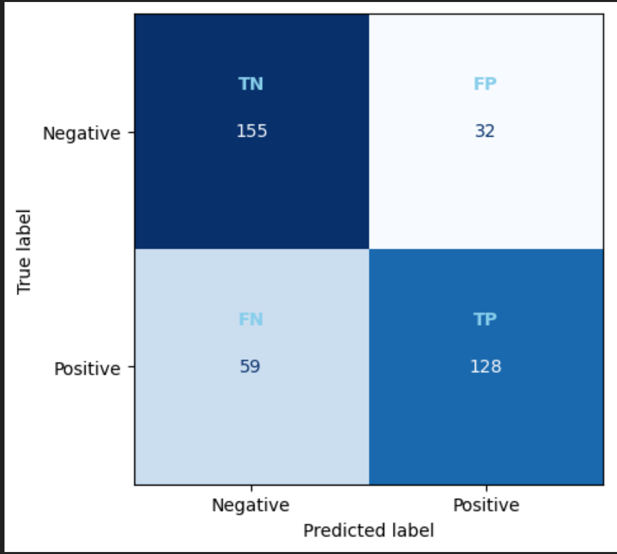
	threshold	f1/5 train	f1/5 test
16	0.76	0.974625	0.856654
12	0.72	0.978815	0.853682
7	0.67	0.965551	0.853458
5	0.65	0.965119	0.852700
2	0.62	0.965354	0.850701

```
0.7948411750656795 No treatment
0.802469135802469 Parameter Tuning
0.8566538296961916 Parameter Tuning + Optimized Threshold
```

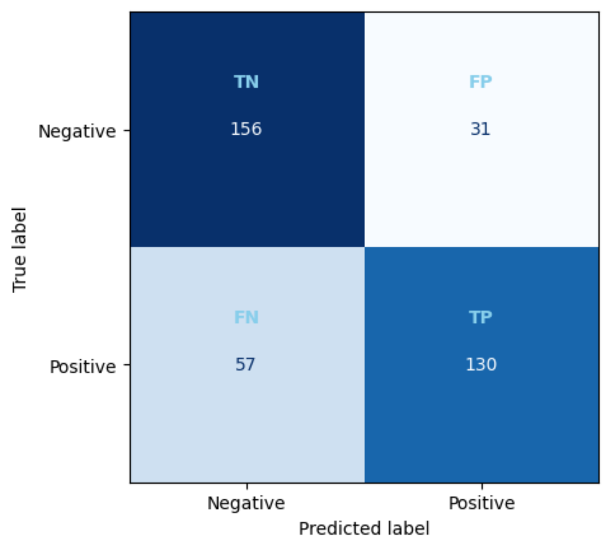
05

Conclusion

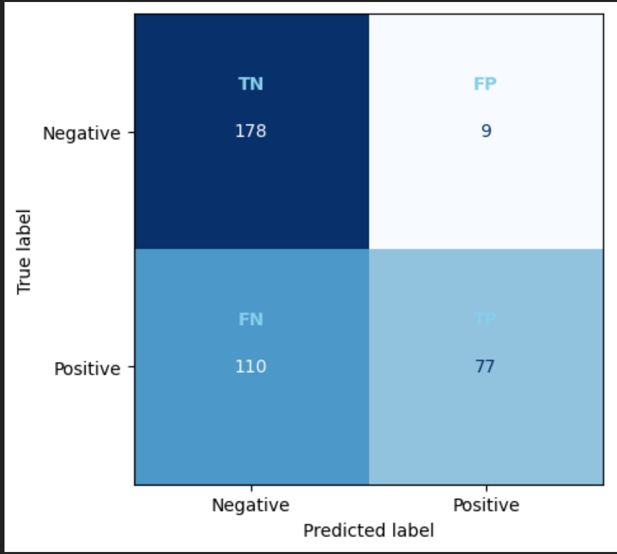
Confusion Matrix



Before Tuning:
- FP:
32 x 4000 SGD = 128000 SGD
- FN:
59 x 800 SGD = 47200 SGD
Total loss: 175200 SGD

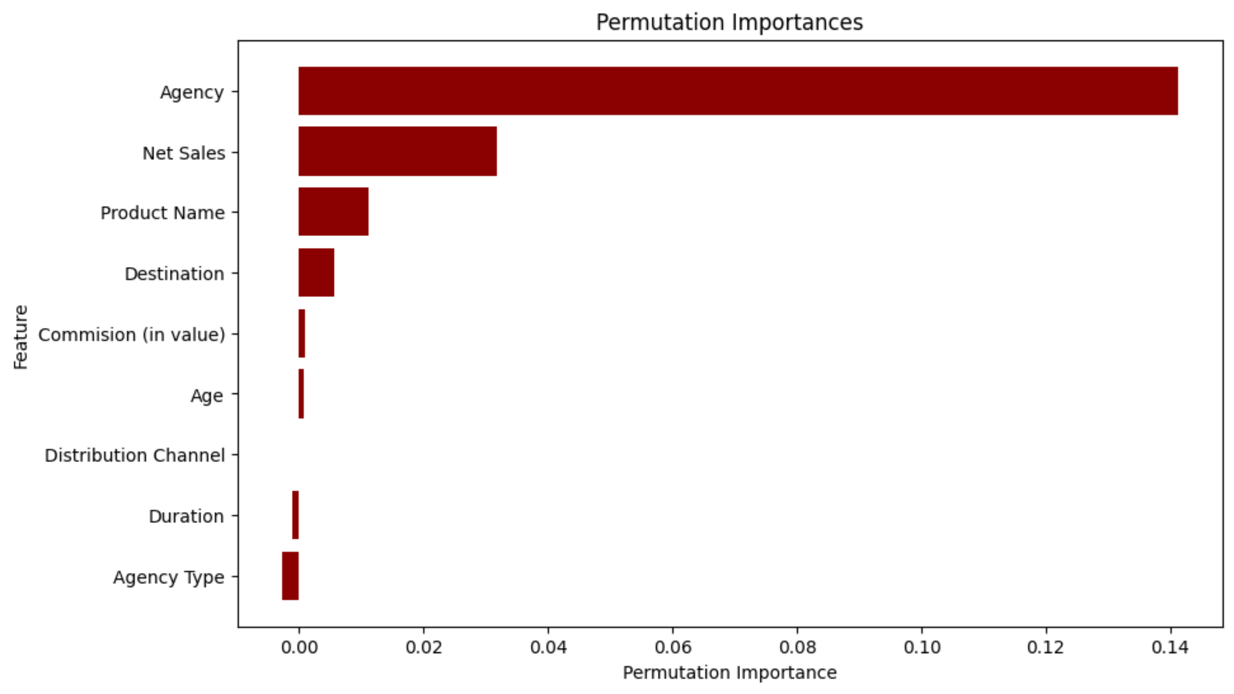


After Tuning:
- FP:
31 x 4000 SGD = 124000 SGD
- FN:
57 x 800 SGD = 45600 SGD
Total loss: 169600 SGD



After Tuning + Threshold:
- FP:
9 x 4000 SGD = 36000 SGD
- FN:
110 x 800 SGD = 88000 SGD
Total loss: 124000 SGD

Feature Importance



Recommendation

06

Recommendations

Recommendations to improve model performance:

- Perform Hyperparameter tuning with GridSearchCV to find the optimal parameter combination in the model.

- Add data to the Claim target in the positive class (Claim = 1).

- Try to handle imbalance using other methods besides Random Under Sampling.

- Try using other metrics such as a combination of Recall and FPR.

Thank You!