Mini-Project 3

Hotel Reservation Prediction

Zwe Min Maw	6238135
Saw Zwe Wai Yan	6318013
Thanarit Kanjanametawat	6410322

Agenda

01 Recap

02 Update

03 Comparison

04 Conclusion

01

Recap

Random Forest

Random Forest:
Accuracy: 0.890818858560794
Precision: 0.880838894184938
Recall: 0.7732217573221757
F1 Score: 0.8235294117647058

Decision-Tree

Decision Tree:
Accuracy: 0.8828232699200441
Precision: 0.8425266903914591
Recall: 0.792468619246862
F1 Score: 0.8167313497197067

Naïve-Bayes

Naive Bayes:
Accuracy: 0.3416046319272126
Precision: 0.33351939715322354
Recall: 1.0
F1 Score: 0.5002092925910423

Neural Network

SVM

Data Processing

Grid Search

Parameter Improvement

02

Update

Data Processing

Discretization

2.1 Lead Time Discretization

In 45
```
# Discretize the lead time in long medium short
# 0-50 short
# 50-100 medium
# 100-1000 long
train_set['lead_time'] = pd.cut(train_set['lead_time'], bins=[0, 50, 100, 1000], labels=['short', 'medium', 'long'])
```
Executed at 2023.09.25 21:35:48 in 90ms

In 46
```
train_set['lead_time'].value_counts()
```
Executed at 2023.09.25 21:35:48 in 91ms

Out 46
```
short     14002
long      10750
medium     6733
Name: lead_time, dtype: int64
```

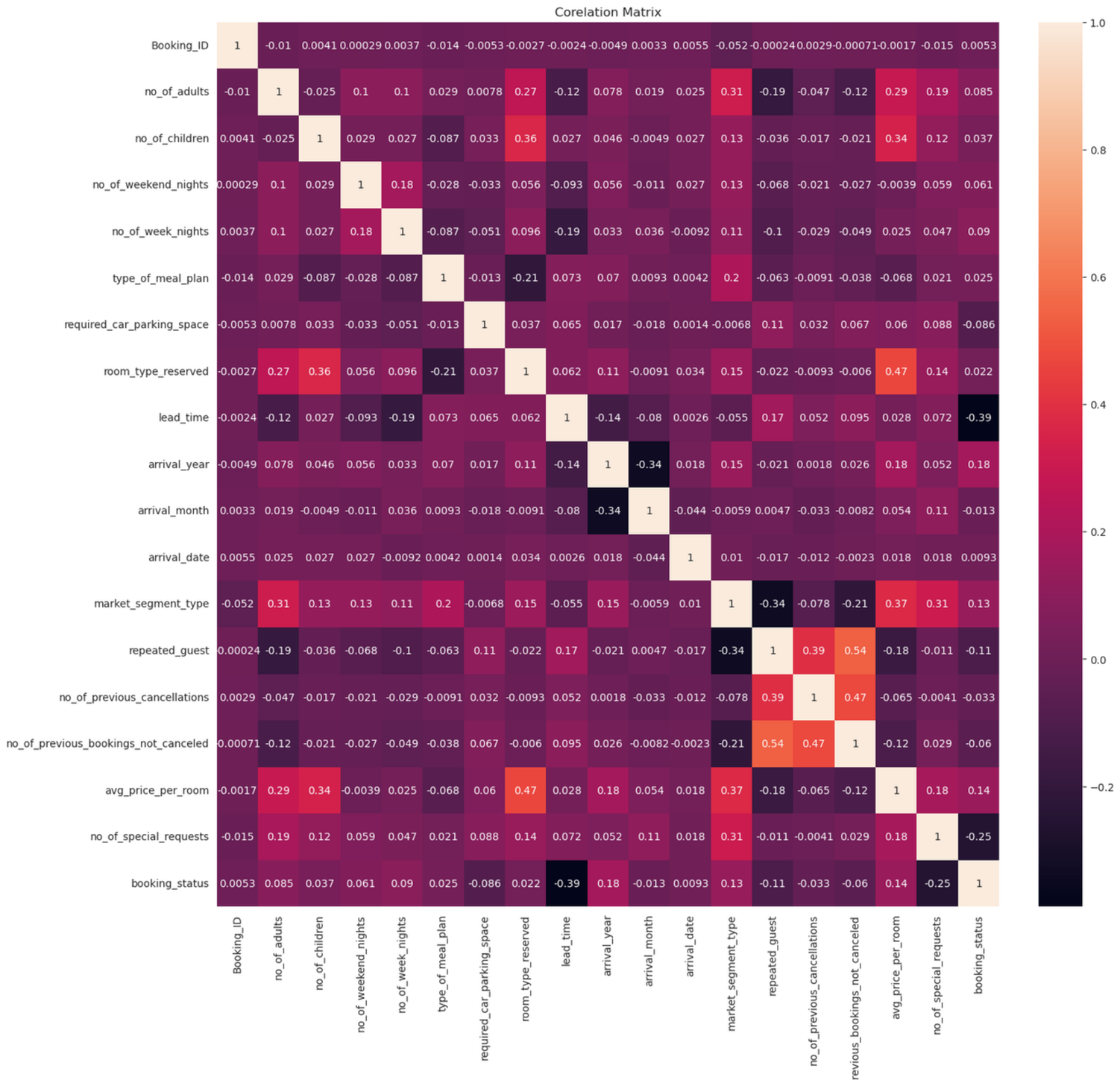
Data Processing

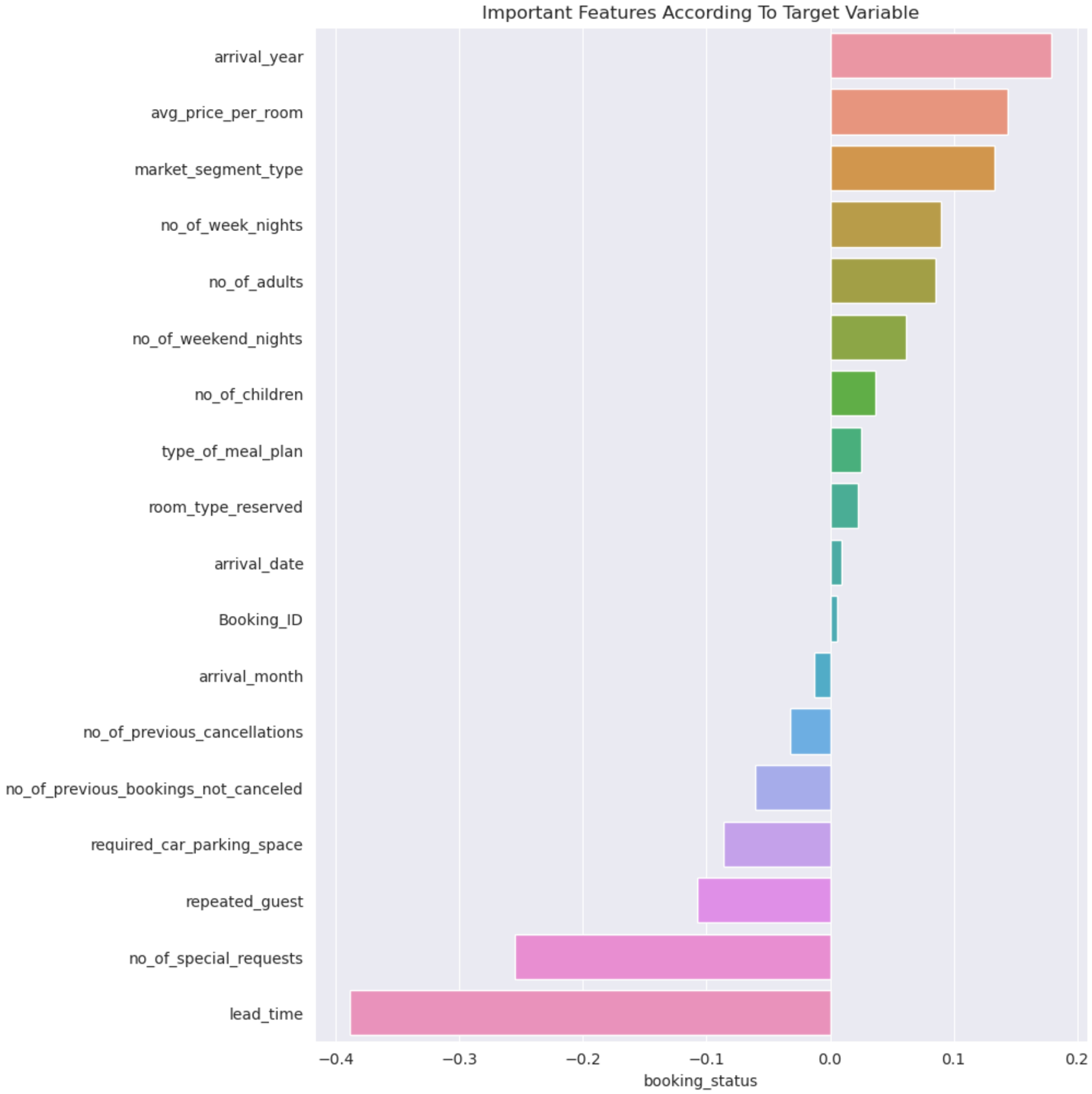
Lable Encoding

```python
# Datatype changing
categorical=train_set.select_dtypes(exclude=[np.number])

for i in categorical.columns:
    train_set[i]=train_set[i].astype('category')

# Data Info

train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32648 entries, 0 to 32647
Data columns (total 19 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Booking_ID                            32648 non-null  category
 1   no_of_adults                          32648 non-null  int64
 2   no_of_children                        32648 non-null  int64
 3   no_of_weekend_nights                  32648 non-null  int64
 4   no_of_week_nights                     32648 non-null  int64
 5   type_of_meal_plan                     32648 non-null  category
 6   required_car_parking_space            32648 non-null  int64
 7   room_type_reserved                    32648 non-null  category
 8   lead_time                             31485 non-null  category
 9   arrival_year                          32648 non-null  int64
 10  arrival_month                         32648 non-null  int64
 11  arrival_date                          32648 non-null  int64
 12  market_segment_type                   32648 non-null  category
 13  repeated_guest                        32648 non-null  int64
 14  no_of_previous_cancellations          32648 non-null  int64
 15  no_of_previous_bookings_not_canceled  32648 non-null  int64
 16  avg_price_per_room                    32648 non-null  float64
 17  no_of_special_requests                32648 non-null  int64
 18  booking_status                        32648 non-null  category
dtypes: category(6), float64(1), int64(12)
```

Correlation Matrix





Correlation between each features and label

Feature Selection for Neural Networks

```
features = target_s.index.tolist()
features.remove('Booking_ID')
features.remove('no_of_special_requests')
features.remove('lead_time')

train_x = train_x[features]
test_x = test_x[features]
```

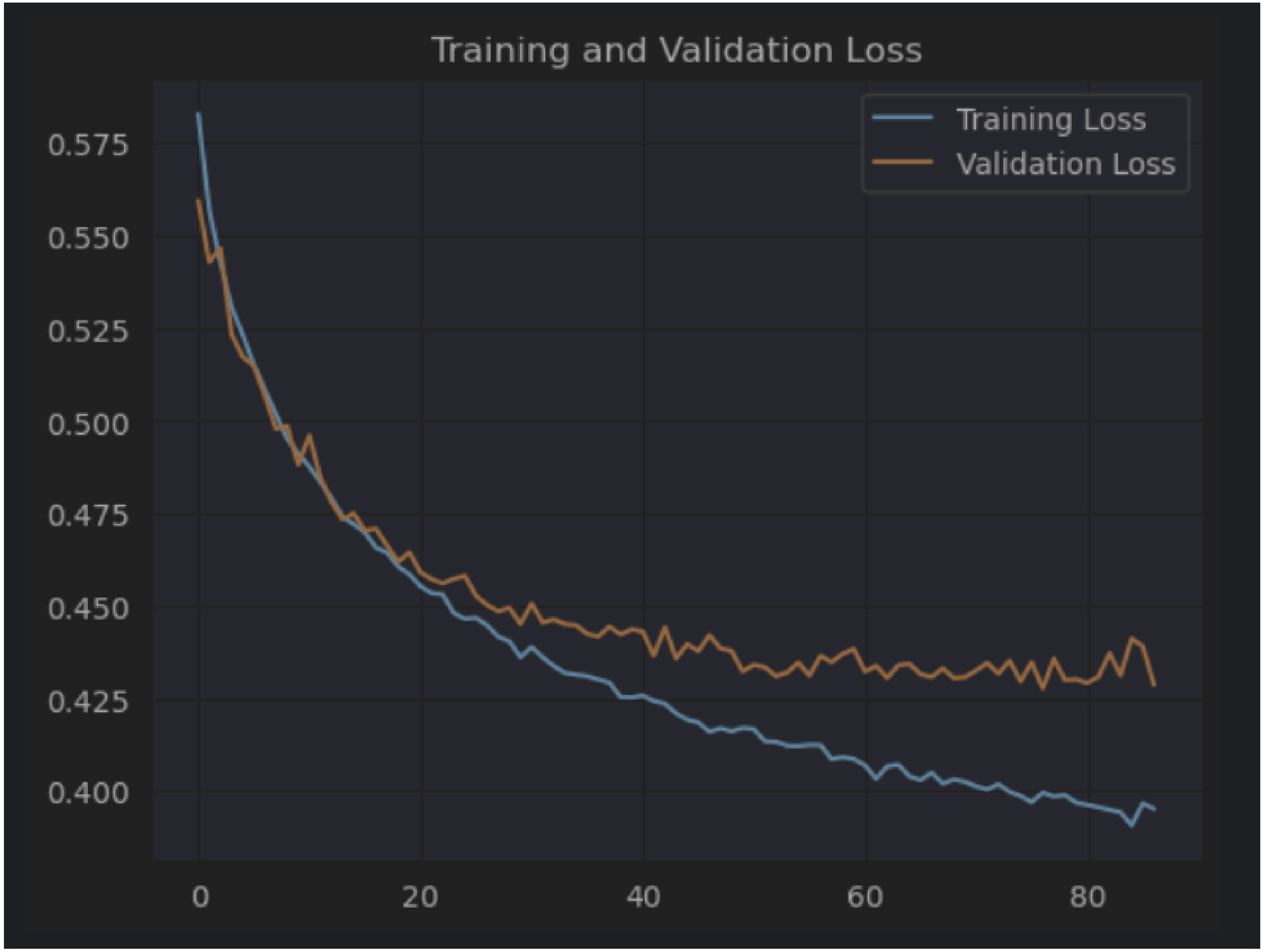
```
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import EarlyStopping

# Early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10)

# Neural Network Model
dl_model = Sequential([keras.layers.Dense(256, activation = "relu",input_shape = [len(train_x.keys())]),
                       keras.layers.Dropout(0.3),

                       keras.layers.Dense(256, activation = "relu"),
                       keras.layers.Dropout(0.3),

                       keras.layers.Dense(256, activation = "relu"),
                       keras.layers.Dropout(0.3),

                       keras.layers.Dense(1, activation="sigmoid")])
# Model Compilation
dl_model.compile(optimizer="adam", loss='binary_crossentropy', metrics=['accuracy'])
result=dl_model.fit(train_x,train_y,
                    validation_data=(test_x,test_y),
                    batch_size=256,
                    epochs=1000,
                    callbacks=[early_stopping])
```

Neural Network

Check if Neural Network is Overfitting

```
import matplotlib.pyplot as plt


plt.plot(result.history['loss'], label='Training Loss')
plt.plot(result.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```

Executed at 2023.09.25 22:40:51 in 321ms


Training and Validation Loss
Training Loss
Validation Loss
0.575
0.550
0.525
0.500
0.475
0.450
0.425
0.400
0
20
40
60
80

Prediction and Evaluation

```
# Predict the classes using the test set (if the probability is greater than 0.5, the class is 1, otherwise 0)
y_pred = dl_model.predict(test_x)
y_pred_classes = (y_pred > 0.5).astype("int32")

# Print Accuracy, Precision, Recall, and F1 Score
print("Neural Network:")
print("Accuracy:", accuracy_score(test_y, y_pred_classes))
print("Precision:", precision_score(test_y, y_pred_classes, average='binary', pos_label=1))
print("Recall:", recall_score(test_y, y_pred_classes, average='binary', pos_label=1))
print("F1 Score:", f1_score(test_y, y_pred_classes, average='binary', pos_label=1))
print("---------------------")
```

Prediction and Evaluation

```
114/114 [==============================] - 0s 1ms/step
Neural Network:
Accuracy: 0.7843948166528811
Precision: 0.720855614973262
Recall: 0.5640167364016736
F1 Score: 0.6328638497652582
----------------------
```

Decision Tree with Grid Search

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

param_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4, 8],
}

grid_search = GridSearchCV(estimator=DecisionTreeClassifier(random_state=3),
                           param_grid=param_grid,
                           cv=5, n_jobs=-1, verbose=2)

grid_search.fit(train_x, train_y)

best_params = grid_search.best_params_
print("Decision Tree's Best parameters:", best_params)
```

Decision Tree with Grid Search

```
Fitting 5 folds for each of 144 candidates, totalling 720 fits
Decision Tree's Best parameters: {'criterion': 'entropy', 'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
```

```
# Initialize and train the Decision Tree Classifier
decision_tree_model = DecisionTreeClassifier(**best_params, random_state=3)
decision_tree_model.fit(train_x, train_y)
```

```
Decision Tree:
Accuracy: 0.8726220016542597
Precision: 0.8051623646960866
Recall: 0.8092050209205021
F1 Score: 0.8071786310517529
----------------------
```

Random Forest with Grid Search

```
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

grid_search = GridSearchCV(
    estimator=RandomForestClassifier(random_state=3),
    param_grid=param_grid,
    cv=5,
    n_jobs=-1,
    verbose=2
)

grid_search.fit(train_x, train_y)

best_params = grid_search.best_params_
print("Random Forest's Best parameters:", best_params)
```

Random Forest with Grid Search

```
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Random Forest's Best parameters: {'max_depth': 30, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
```

```python
# Initialize and train the Random Forest Classifier
random_forest_model = RandomForestClassifier(**best_params, random_state=3)
random_forest_model.fit(train_x, train_y)
```

```
Random Forest:
Accuracy: 0.9018472566859663
Precision: 0.8817106460418562
Recall: 0.8108786610878661
F1 Score: 0.8448125544899738
----------------------
```

SVM with Grid Search

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Define hyperparameter grid
param_grid = {
    'C': [1, 10],
    'kernel': ['linear'],
    'gamma': [1, 10]
}

# Initialize GridSearchCV
grid_search = GridSearchCV(SVC(random_state=3), param_grid, cv=5, scoring='accuracy', n_jobs=-1,verbose=2)

# Fit to the training data
grid_search.fit(train_x, train_y)

# Retrieve the best model
best_params = grid_search.best_params_
```

SVM with Grid Search

```
{'C': 10, 'gamma': 1, 'kernel': 'linear'}
```

```
svm_model = SVC(**best_params, random_state=3)
svm_model.fit(train_x, train_y)
```

```
Support Vector Machine (SVM):
Accuracy: 0.8067273228563551
Precision: 0.7282809611829945
Recall: 0.6594142259414226
F1 Score: 0.6921387790953009
----------------------------
```

Naive Bayes

```python
# Initialize and train the Naive Bayes Classifier
naive_bayes_model = GaussianNB()
naive_bayes_model.fit(train_x, train_y)

# Predict on the test dataset for each model
y_pred_naive_bayes = naive_bayes_model.predict(test_x)

# Calculate evaluation metrics for each model
accuracy_naive_bayes = accuracy_score(test_y, y_pred_naive_bayes)
precision_naive_bayes = precision_score(test_y, y_pred_naive_bayes, average='binary', pos_label=1)
recall_naive_bayes = recall_score(test_y, y_pred_naive_bayes, average='binary', pos_label=1)
f1_naive_bayes = f1_score(test_y, y_pred_naive_bayes, average='binary', pos_label=1)

# Print evaluation metrics for each model
print("Naive Bayes:")
print("Accuracy:", accuracy_naive_bayes)
print("Precision:", precision_naive_bayes)
print("Recall:", recall_naive_bayes)
print("F1 Score:", f1_naive_bayes)
print("----------------------")
```

```
Accuracy: 0.41494347945960847
Precision: 0.35707678075855687
Recall: 0.9690376569037656
F1 Score: 0.5218566922036953
----------------------
```

03

Comparison

Evaluation Comparison

	Accuracy	Precision	Recall	F1-Measure	Overall
Random Forest	Progress : 0.890818 Final: 0.90184	Progress : 0.88083 Final: 0.88171	Progress : 0.77322 Final: 0.81087	Progress : 0.82352 Final: 0.84481	**Better**
Decision Tree	Progress : 0.88282 Final: 0.87262	Progress : 0.84252 Final: 0.80516	Progress : 0.79246 Final: 0.80920	Progress : 0.81673 Final: 0.80871	**Worse**
Naive Bayes	Progress : 0.34160 Final: 0.4194	Progress : 0.33351 Final: 0.35707	Progress : 1.0 Final: 0.96903	Progress : 0.50020 Final: 0.52185	**Better**
SVM	Progress : null Final: 0.80672	Progress : null Final: 0.72828	Progress : null Final: 0.65941	Progress : null Final: 0.69213	-
Neural Network	Progress : null Final: 0.78439	Progress : null Final: 0.72085	Progress : null Final: 0.56401	Progress : null Final: 0.63286	-

04

Conclusion

Random Forest

- Among all the models, including the new models, Random Forest outperformed them in terms of evaluation.
- The evaluation results of the new model, with grid selection, are better than the previous output, achieving an accuracy of 90%.

Thus, we can conclude that the Random Forest is the best model for this problem.

Thank You
For Your Attention