Before performing different machine learning models, the “customerID” column is dropped because this column serves as an identifier for each data point instead of a feature. Therefore, “cuscomerID” column will not be included as a feature in any models.

The label that the model is developed to predict is the “Churn” column. Three tree-based models are explored in this section, namely simple decision tree, random forest and AdaBoost method. In each case, the data set is splitted into a train set and a test set. A grid search and cross validation will be applied to the train set during training the model, a combination of parameters that gives the best prediction will be chosen for the model.

A note in the grid search procedure. The set of parameters is chosen at relatively coarse scale for the first time fitting the data. The best combination of parameters is printed out after fitting, then new set of parameters with finer scale around the best values are chosen for the next fitting. This procedure is repeated until the best parameters are determined. The python code snippets shown below represent the last iteration.

1. Simple decision tree model

Python code:

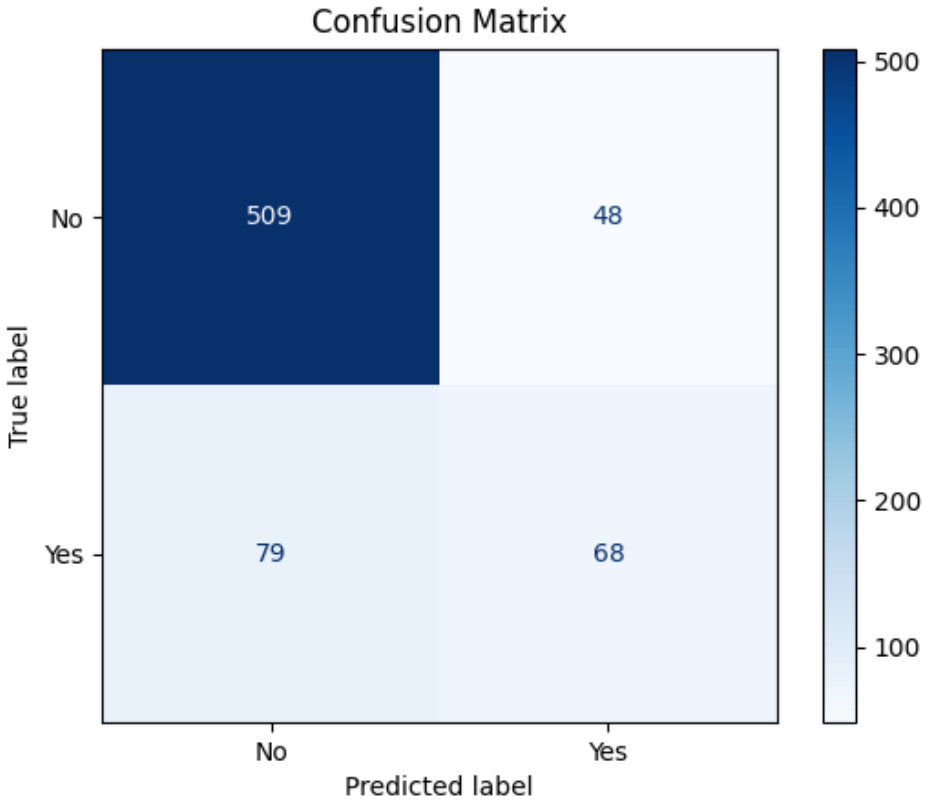
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report,
ConfusionMatrixDisplay

df = pd.read_csv('Telco-Customer-Churn.csv')
X = pd.get_dummies(df.drop(['customerID', 'Churn'], axis=1),
drop_first=True)
y = df['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.1, random_state=101)

model = DecisionTreeClassifier(random_state=101)
param_grid = {'max_depth': list(range(2, 8)),
              'max_leaf_nodes': list(range(17, 23)),
              'max_features': list(range(15, 20))}
grid = GridSearchCV(estimator=model, param_grid=param_grid)
grid.fit(X_train, y_train)
best_params = grid.best_params_
print(best_params)
best_model = DecisionTreeClassifier(**best_params)
best_model.fit(X_train, y_train)
```

```
prediction = best_model.predict(X_test)
confusion = confusion_matrix(y_test, prediction)
disp = ConfusionMatrixDisplay(confusion_matrix=confusion,
                        display_labels=['No', 'Yes'])
disp.plot(cmap='Blues')
plt.title('Confusion Matrix')
plt.show()
print(classification_report(y_test, prediction))
feature_importance = pd.DataFrame(index=X.columns,
data=best_model.feature_importances_, columns=['Importance'])\
    .sort_values(by='Importance')
high_importance = feature_importance[feature_importance >
0.01].dropna()
print(high_importance)
```

Result:



- Hyper parameters:

'max_depth': 6, 'max_features': 18, 'max_leaf_nodes': 20

- Classification Report

	precision	recall	f1-score	support
No	0.86	0.89	0.88	557
Yes	0.52	0.44	0.48	147
accuracy			0.80	704
macro avg	0.69	0.67	0.68	704
weighted avg	0.79	0.80	0.79	704

- Feature importance (importance less than 0.01 are omitted):

	Importance
MonthlyCharges	0.010224
Contract_Two year	0.030923
Contract_One year	0.031538
DeviceProtection_No internet service	0.036980
TotalCharges	0.048878
PaymentMethod_Electronic check	0.102840
InternetService_Fiber optic	0.292605
Tenure	0.426672

2. Random forest model

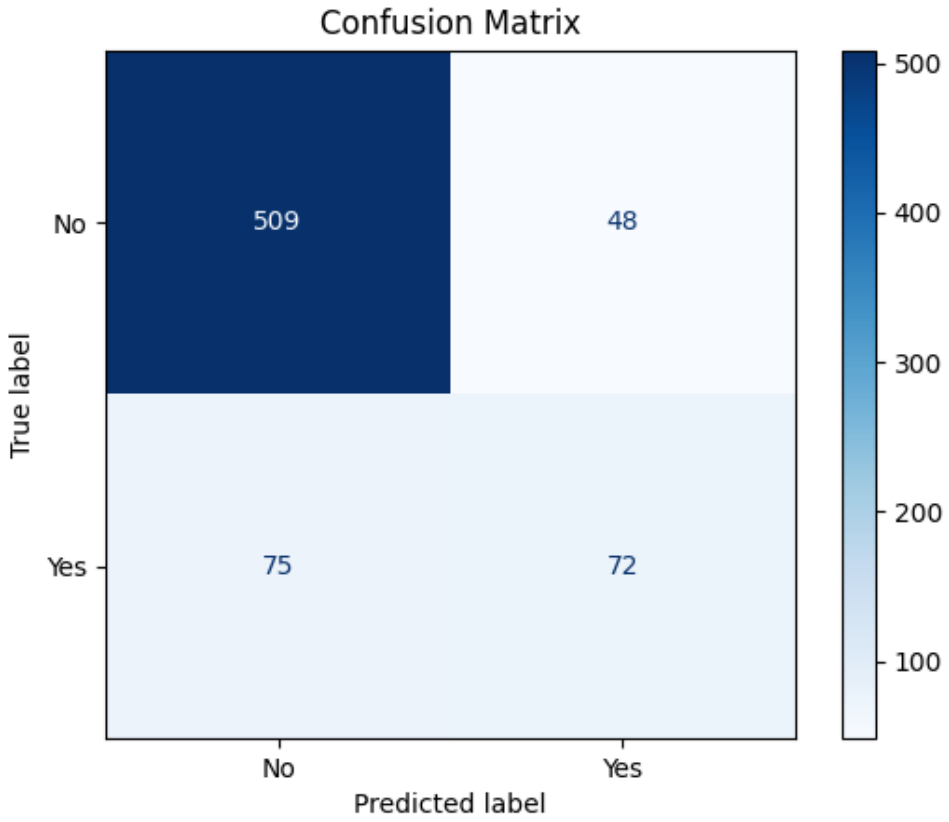
Python code:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
classification_report

df = pd.read_csv('Telco-Customer-Churn.csv')
X = pd.get_dummies(df.drop(['customerID', 'Churn'], axis=1),
drop_first=True)
y = df['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.1, random_state=101)

model = RandomForestClassifier(max_features='sqrt', bootstrap=True,
random_state=101)
param_grid = {'n_estimators': list(range(75, 82)),
              'max_depth': list(range(6, 12))}
grid = GridSearchCV(estimator=model, param_grid=param_grid)
grid.fit(X_train, y_train)
best_params = grid.best_params_
print(best_params)
best_model = RandomForestClassifier(**best_params)
best_model.fit(X_train, y_train)
prediction = best_model.predict(X_test)
confusion = confusion_matrix(y_test, prediction)
disp = ConfusionMatrixDisplay(confusion_matrix=confusion,
                              display_labels=['No', 'Yes'])
disp.plot(cmap='Blues')
plt.title('Confusion Matrix')
plt.show()
print(classification_report(y_test, prediction))
feature_importance = pd.DataFrame(index=X.columns,
data=best_model.feature_importances_, columns=['Importance'])\
    .sort_values(by='Importance')
high_importance = feature_importance[feature_importance >
0.01].dropna()
print(high_importance)
```

Result



- Hyper parameters:

'n_estimators': 80, 'max_depth': 8

- Classification Report:

	precision	recall	f1-score	support
No	0.87	0.92	0.90	557
Yes	0.62	0.49	0.55	147
accuracy			0.83	704
macro avg	0.74	0.70	0.72	704
weighted avg	0.82	0.83	0.82	704

- Feature importance (importance less than 0.01 are omitted):

	Importance
InternetService_No	0.011172
SeniorCitizen	0.011245
OnlineBackup_No internet service	0.011584
OnlineBackup_Yes	0.012527
OnlineSecurity_No internet service	0.013377
DeviceProtection_No internet service	0.013770
StreamingTV_No internet service	0.014458
PaperlessBilling_Yes	0.019694
StreamingMovies_No internet service	0.022775
TechSupport_Yes	0.024486
OnlineSecurity_Yes	0.035750
Contract_One year	0.042391
PaymentMethod_Electronic check	0.069132
Contract_Two year	0.084580
InternetService_Fiber optic	0.084718
MonthlyCharges	0.089614
TotalCharges	0.154129
Tenure	0.201595

3. AdaBoost model

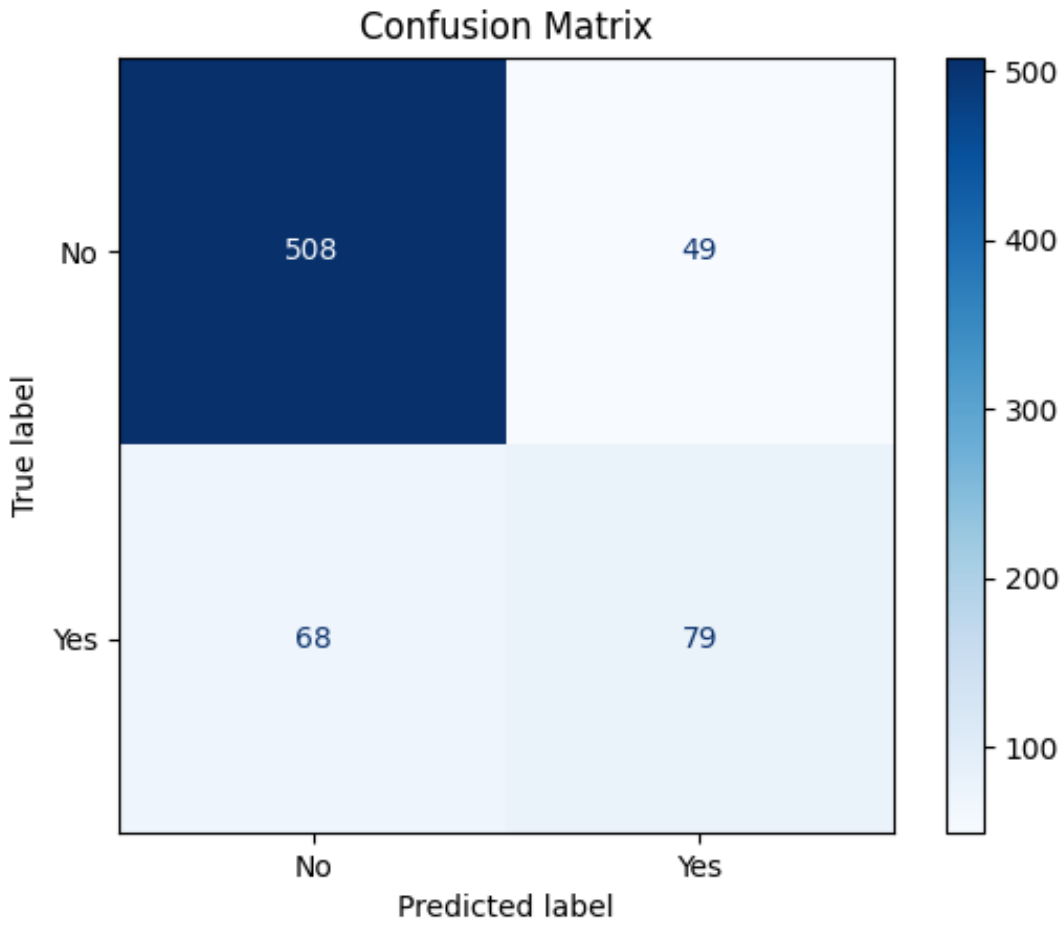
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
classification_report

df = pd.read_csv('Telco-Customer-Churn.csv')
X = pd.get_dummies(df.drop(['customerID', 'Churn'], axis=1),
drop_first=True)
y = df['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.1, random_state=101)

model = AdaBoostClassifier(random_state=101)
param_grid = {'n_estimators': list(range(50, 60))}
grid = GridSearchCV(estimator=model, param_grid=param_grid)
grid.fit(X_train, y_train)
best_params = grid.best_params_
print(best_params)
best_model = AdaBoostClassifier(**best_params)
best_model.fit(X_train, y_train)
prediction = best_model.predict(X_test)
confusion = confusion_matrix(y_test, prediction)
disp = ConfusionMatrixDisplay(confusion_matrix=confusion,
                              display_labels=['No', 'Yes'])
disp.plot(cmap='Blues')
plt.title('Confusion Matrix')
plt.show()
print(classification_report(y_test, prediction))
feature_importance = pd.DataFrame(index=X.columns,
data=best_model.feature_importances_, columns=['Importance'])\
    .sort_values(by='Importance')
high_importance = feature_importance[feature_importance >
0.01].dropna()
print(high_importance)
```

- Hyper parameters:

'n_estimators': 52. The base estimator used in the AdaBoost model is DecisionTreeClassifier with depth equal to 1 (stump).



- Classification Report:

	precision	recall	f1-score	support
No	0.88	0.91	0.90	557
Yes	0.62	0.54	0.57	147
accuracy			0.83	704
macro avg	0.75	0.72	0.74	704
weighted avg	0.83	0.83	0.83	704

- Feature importance (importance less than 0.01 are omitted):

	Importance
StreamingTV_Yes	0.019231
Contract_One year	0.019231
TechSupport_Yes	0.019231
PaperlessBilling_Yes	0.019231
StreamingMovies_Yes	0.019231
StreamingMovies_No internet service	0.019231
SeniorCitizen	0.019231
OnlineSecurity_Yes	0.019231
PaymentMethod_Electronic check	0.019231
PaymentMethod_Mailed check	0.019231
MultipleLines_Yes	0.038462
PhoneService_Yes	0.038462
Contract_Two year	0.038462
InternetService_Fiber optic	0.057692
MonthlyCharges	0.153846
Tenure	0.192308
TotalCharges	0.288462

DISCUSSION

The analysis of feature importance yields intriguing insights. The features 'tenure,' 'monthly charges,' and 'total charges' emerge as prominent factors, occupying high positions in the ranking of influential features. Remarkably, this aligns seamlessly with our previous observations from the exploratory data analysis. It is worth noting, nevertheless, that 'internet service_fiber optic' also emerges with considerable significance in the feature importance ranking. Surprisingly, this attribute, which was relatively overlooked during our earlier discussions, surfaces as a contributor of noteworthy impact according to the feature importance analysis.

Among the three tree-based models (simple decision tree, random forest and AdaBoost model), the simple decision tree exhibits the poorest overall performance. On the other hand, the AdaBoost model demonstrates the most superior overall performance. However, it is important to highlight that the distinctions observed among the aforementioned models do not hold statistical significance. This is evident from the overall accuracy values, which range modestly from 0.80 to 0.83. While the overall accuracy provides a reasonable performance overview, our primary focus lies on the recall score for the 'Yes' class. Regrettably, this recall score fails to meet our expectations across all three models. The subpar recall score for the 'Yes' class reflects a notable deficiency in the models' ability to accurately identify positive instances, inevitably leading to an excessive count of false negative predictions.

To summarize, the tree-based models offer satisfactory performance in terms of overall predictions. However, users employing these models should exercise caution when interpreting negative predictions. These models have demonstrated a tendency to generate an elevated number of false negative predictions, warranting heightened vigilance in such scenarios