Bala Vignesh Bala Subramanian

Project

Predict the default of a credit card

The training data set includes a binary variable, default payment (Yes = 1, No = 0), as the target variable, and the following 23 variables as the features variables:

- X1: Amount of the given credit
- X2: Gender (1 = male; 2 = female)
- X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others)
- X4: Marital status (1 = married; 2 = single; 3 = others)
- X5: Age
- X6 - X11: History of past payment. We tracked the past monthly payment records (from April to September, 2005) as follows: X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005; . . .;X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.
- X12-X17: Amount of bill statement. X12 = amount of bill statement in September, 2005; X13 = amount of bill statement in August, 2005; . . .; X17 = amount of bill statement in April, 2005.
- X18-X23: Amount of previous payment. X18 = amount paid in September, 2005; X19 = amount paid in August, 2005; . . .;X23 = amount paid in April, 2005.

Hint: Check the target variable; is the classes balanced or imbalanced?

data file: https://raw.githubusercontent.com/franklin-univ-data-science/data/master/credit_default.csv

Actions

Implement a model in Jupyter Notebook and discuss the following topics:

- Describe the problem
  - What is the problem?
  - What is the type of machine learning?
  - What are the feature variables and target variables?
- Data exploration and preprocessing

    - How did you explore the data?
    - How did you clean the data (are there missing or invalid values)?
- Modeling
    - Split 20% data as the test set using the random status 123.
    - What machine learning algorithms were used? Which is better?
    - What evaluation metric do you prefer?
    - How did you evaluation model's performance?
    - How did you diagnose the model? Is it overfitting, under fitting, or good fitting?
- Results and discussion
    - What is your model's results? Is it good? Do you have any concerns?

Due Date: the last week.

```
import pandas as pd

data = pd.read_csv('https://raw.githubusercontent.com/franklin-univ-data-science/da
```

Problem Description

The problem given is to find the who will be on a Credit card default list, This problem can be solved via either supervised learning like Decistion tree or random forest or unsupervised learning like KNN, we can used all the available variables as feature variables except ID columns and target varible is Y as either 1 or 0

Data exploration and preprocessing

```
data.head()
```

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	
0	1	20000	2	2	1	24	2	2	-1	-1	-2	-2	3913	3102	689	0	
1	2	120000	2	2	2	26	-1	2	0	0	0	2	2682	1725	2682	3272	3
2	3	90000	2	2	2	34	0	0	0	0	0	0	29239	14027	13559	14331	14
3	4	50000	2	2	1	37	0	0	0	0	0	0	46990	48233	49291	28314	28
4	5	50000	1	2	1	57	-1	0	-1	0	0	0	8617	5670	35835	20940	19

```
1 data.shape
```

```
(30000, 25)
```

```
1 data.describe()
```

	ID	X1	X2	X3	X4	X5	X6	X'
count	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.00	30,000.0
mean	15,000.50	167,484.32	1.60	1.85	1.55	35.49	-0.02	-0.1
std	8,660.40	129,747.66	0.49	0.79	0.52	9.22	1.12	1.2
min	1.00	10,000.00	1.00	0.00	0.00	21.00	-2.00	-2.0
25%	7,500.75	50,000.00	1.00	1.00	1.00	28.00	-1.00	-1.0
50%	15,000.50	140,000.00	2.00	2.00	2.00	34.00	0.00	0.0
75%	22,500.25	240,000.00	2.00	2.00	2.00	41.00	0.00	0.0
max	30,000.00	1,000,000.00	2.00	6.00	3.00	79.00	8.00	8.0

```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      30000 non-null  int64
 1   X1      30000 non-null  int64
 2   X2      30000 non-null  int64
 3   X3      30000 non-null  int64
 4   X4      30000 non-null  int64
 5   X5      30000 non-null  int64
 6   X6      30000 non-null  int64
 7   X7      30000 non-null  int64
 8   X8      30000 non-null  int64
 9   X9      30000 non-null  int64
 10  X10     30000 non-null  int64
 11  X11     30000 non-null  int64
 12  X12     30000 non-null  int64
 13  X13     30000 non-null  int64
 14  X14     30000 non-null  int64
 15  X15     30000 non-null  int64
 16  X16     30000 non-null  int64
 17  X17     30000 non-null  int64
 18  X18     30000 non-null  int64
 19  X19     30000 non-null  int64
 20  X20     30000 non-null  int64
 21  X21     30000 non-null  int64
```

```
 22  X22     30000 non-null  int64
 23  X23     30000 non-null  int64
 24  Y       30000 non-null  int64
dtypes: int64(25)
memory usage: 5.7 MB
```

```
# Remove the ID Column
data=data.drop(['ID'], axis=1)
```

```
# Check Duplicate

data.duplicated().sum()
```

```
35
```

```
# Remove Duplicate
data=data.drop_duplicates()
```

```

data.shape
```

```
(29965, 24)
```

```
#Check Na
data.isna().sum()
```

```
X1     0
X2     0
X3     0
X4     0
X5     0
X6     0
X7     0
X8     0
X9     0
X10    0
X11    0
X12    0
X13    0
X14    0
X15    0
X16    0
X17    0
X18    0
X19    0
X20    0
X21    0
X22    0
X23    0
Y      0
dtype: int64
```

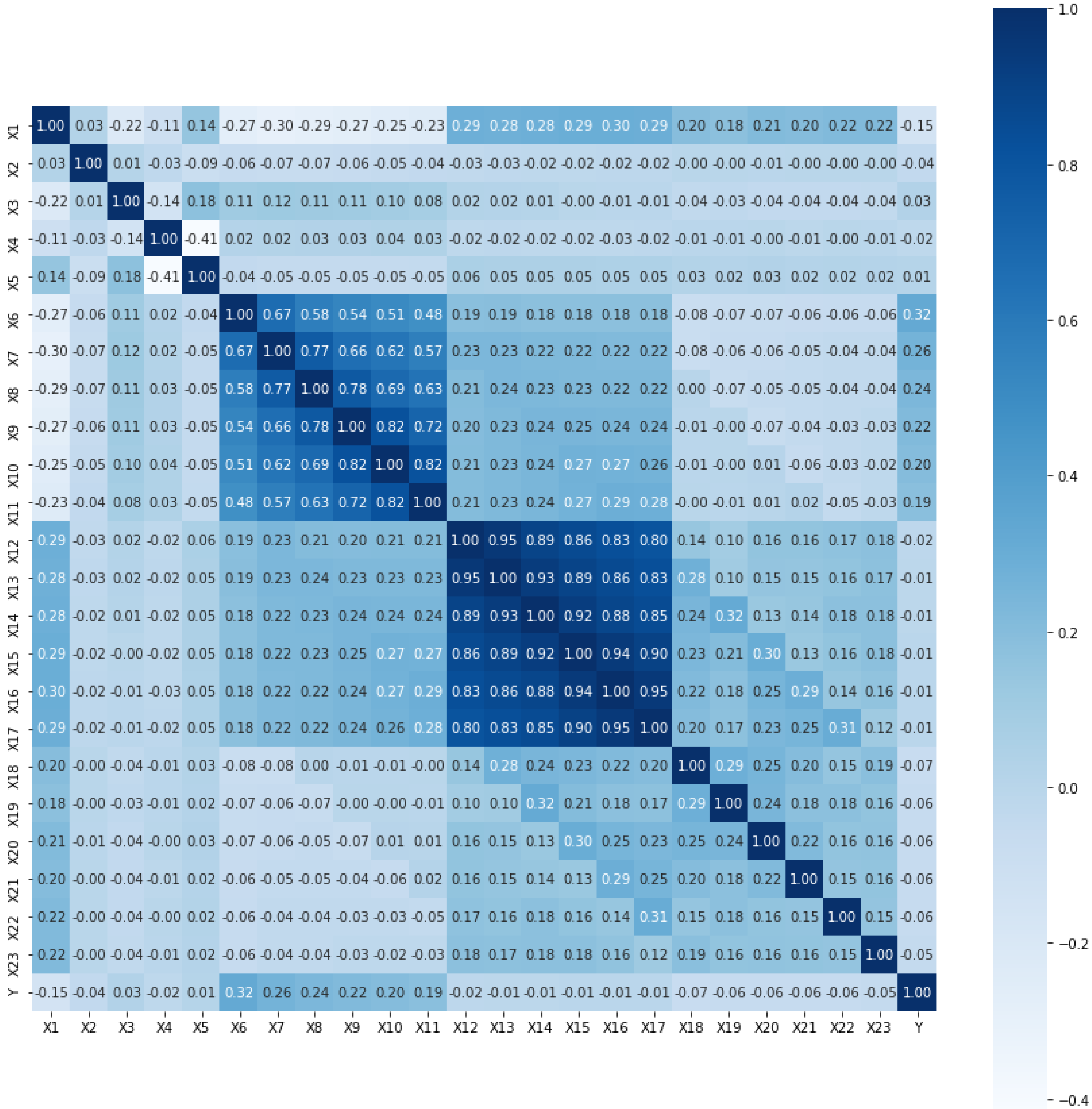
```
data.isnull().sum()
```

```
X1     0
X2     0
X3     0
X4     0
X5     0
X6     0
X7     0
X8     0
X9     0
X10    0
X11    0
X12    0
X13    0
X14    0
X15    0
X16    0
X17    0
X18    0
X19    0
X20    0
X21    0
X22    0
X23    0
Y      0
dtype: int64
```

No Missing or invalid values in the dataset. with the min and max value for education it seems it is not with in 1-4, but i am not saniting the model

Modeling

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

plt.figure(figsize=(15,15))
cm = np.corrcoef(data.values.T)
hm = sns.heatmap(cm,
                 annot=True,
                 square=True,
                 fmt='.2f',
                 yticklabels=data.columns,
                 xticklabels=data.columns,cmap='Blues')


plt.show()
```



from the heat map, there seems to be Y value is highly correlated to X6-X11

```
X, y = data.iloc[:, 0:-1].values, data.iloc[:, -1].values

print(X.shape)
```

```
print(y.shape)
```

```
(29965, 23)
(29965,)
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y
```

```
# Checking Standard Scalar

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)
```

```
# Checking Robust Scalar

from sklearn.preprocessing import RobustScaler

rb = RobustScaler()
X_train_rbt = rb.fit_transform(X_train)
X_test_rbt = rb.transform(X_test)
```

```
from sklearn.decomposition import PCA

pca = PCA()
X_train_pca = pca.fit_transform(X_train_std)
print(pca.explained_variance_ratio_)

X_train_pca = pca.fit_transform(X_train_rbt)
print(pca.explained_variance_ratio_)
```

```
[0.28349316 0.17852747 0.0659031  0.05968878 0.04469172 0.04198665
 0.04004275 0.03872627 0.03850717 0.03696978 0.03367541 0.02942801
 0.02509079 0.02272187 0.01753458 0.01119192 0.01078587 0.00831615
 0.00579386 0.00301313 0.00179704 0.0011071  0.0010074 ]
[2.95814801e-01 1.56893437e-01 1.17477906e-01 1.09680490e-01
 9.78887405e-02 9.56171366e-02 6.10346583e-02 3.04407587e-02
 6.86806566e-03 5.11967149e-03 4.04198565e-03 3.54140589e-03
 2.76464269e-03 2.62370079e-03 2.00792331e-03 1.96266284e-03
 1.76399988e-03 1.35613445e-03 1.31460276e-03 7.90666969e-04
 4.56217726e-04 2.91917079e-04 2.48474473e-04]
```

By seeing the PCA variance, we can try to see if we can use the rubust scalar with PCA of 15 features

Planning to analyze the data using 4 models and going to compare against each other, Models going to use is

1. LogisticRegression
2. KNeighbors
3. DecisionTree
4. RandomForest

   I am planning to use the f1 metrics as the perfomance.

   Model Performance will be based on the Confusion matrix and ROC curve

Since planning to use the Scalar, Cross validation and regularization via grid search, no over or under fitting will be there i guess. we can check the Accuracy score of the traning data cross validation score and test score.

LogisticRegression Analysis

```
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline

pipe_lr = make_pipeline(RobustScaler(),
                        PCA(n_components=15),
                        LogisticRegression(random_state=1, solver='liblinear', pena

pipe_lr.fit(X_train, y_train)
y_pred = pipe_lr.predict(X_test)
print('Test Accuracy: %.3f' % pipe_lr.score(X_test, y_test))
```

```
Test Accuracy: 0.811
```

```
from sklearn.model_selection import GridSearchCV

LRparam_grid = {
    'logisticregression__C': [0.001, 0.01, 0.1, 1, 10]
}

gs = GridSearchCV(estimator=pipe_lr,
                  param_grid=LRparam_grid,
                  scoring='f1',
                  cv=10,verbose=0)

gs = gs.fit(X_train, y_train)
print(gs.best_score_)
```

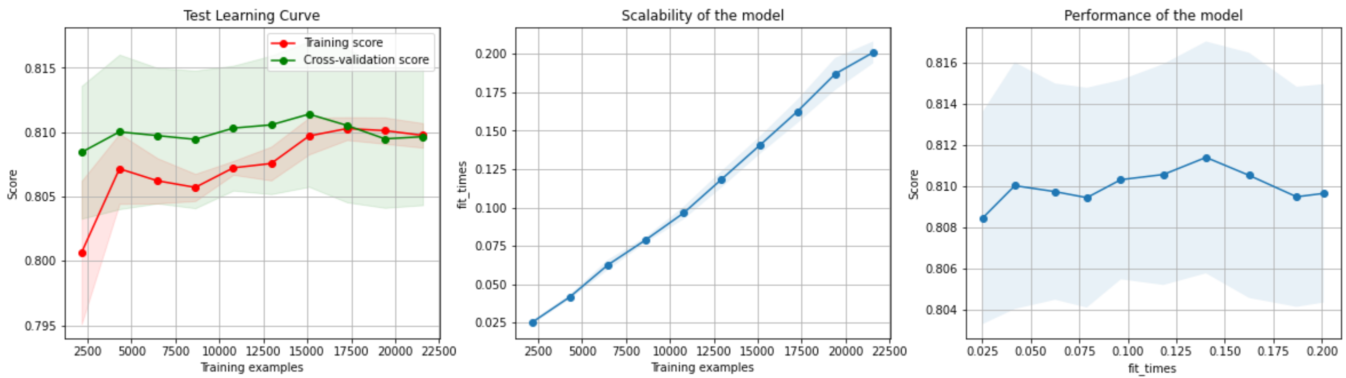
```
print(gs.best_params_)
```

```
0.36911759528791566
{'logisticregression__C': 0.001}
```

Run plot_learning_curve Function

```
plot_learning_curve(gs.best_estimator_, 'Test Learning Curve', X_train, y_train, a
                        n_jobs=None, train_sizes=np.linspace(.1, 1.0, 10))
```

```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.6/dist-packages/matplot
```



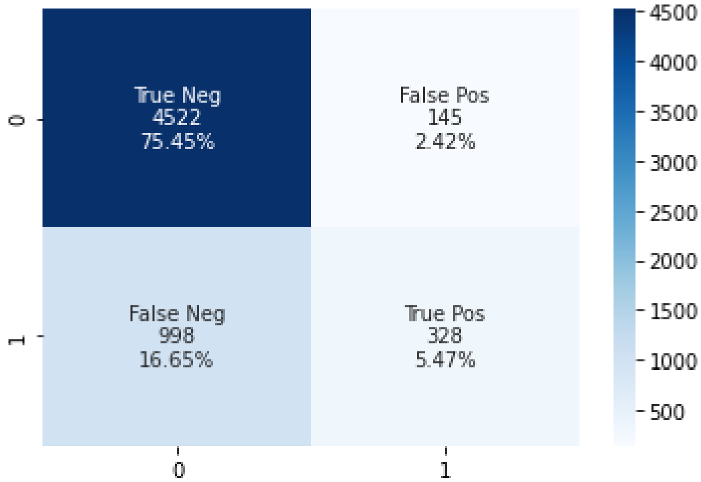

```
clf_lr = gs.best_estimator_ # pick the best model
clf_lr.fit(X_train, y_train)
test_score_lr = '%.3f' % clf_lr.score(X_test, y_test)
print(f'Test accuracy: {test_score_lr}')
```

```
Test accuracy: 0.809
```

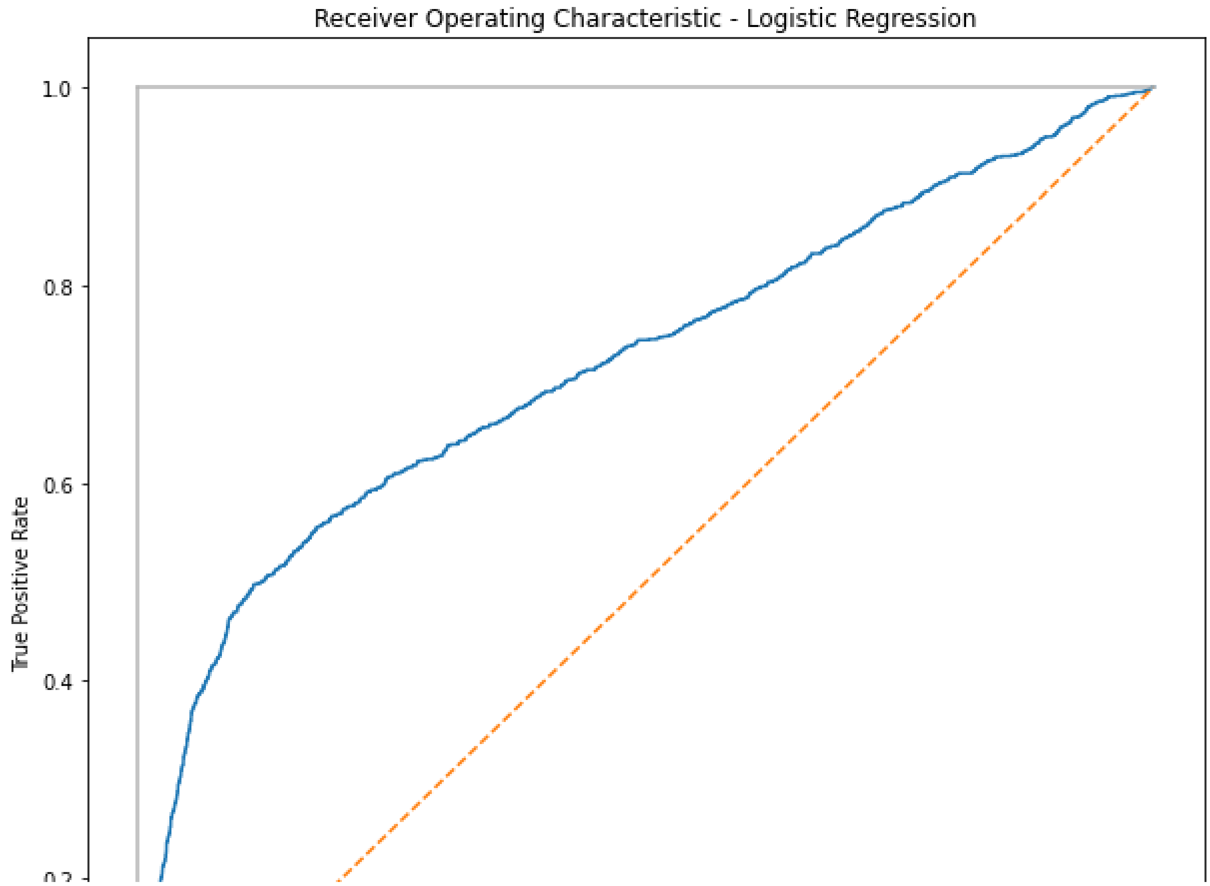
```
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_sc

y_pred_lr = clf_lr.predict(X_test)
confmat = confusion_matrix(y_true=y_test, y_pred=y_pred_lr)

group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                confmat.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     confmat.flatten()/np.sum(confmat)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
```

```
sns.heatmap(confmat, annot=labels, fmt='', cmap='Blues')

print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred_lr))
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred_lr))
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred_lr))
```

```
Precision: 0.693
Recall: 0.247
F1: 0.365
```




```
##Computing false and true positive rates
from sklearn.metrics import roc_curve, roc_auc_score

# Get predicted probabilities
y_score_lr = clf_lr.predict_proba(X_test)[:,1]

false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, y_score_lr)

# Plot ROC curves
plt.subplots(1, figsize=(10,10))
plt.title('Receiver Operating Characteristic - Logistic Regression')
plt.plot(false_positive_rate, true_positive_rate, label='Logistic Regression')
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc='lower right')
plt.show()
```



KNN Analysis

```
from sklearn.neighbors import KNeighborsClassifier

pipe_knn = make_pipeline(RobustScaler(),
                         PCA(n_components=15),
                         KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski'

pipe_knn.fit(X_train, y_train)
y_pred = pipe_knn.predict(X_test)
print('Test Accuracy: %.3f' % pipe_knn.score(X_test, y_test))
```

```
Test Accuracy: 0.801
```

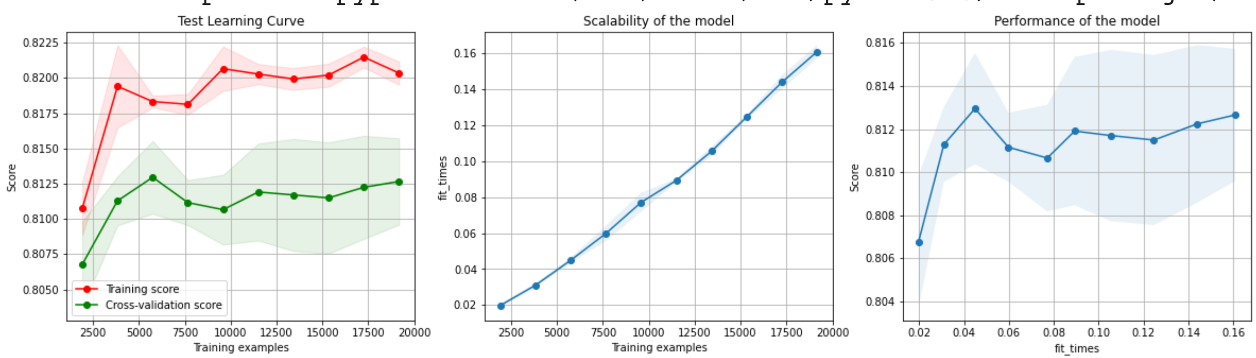
```
KNNparam_grid = {
    'kneighborsclassifier__n_neighbors': range(3,30)
}

gs_knn = GridSearchCV(estimator=pipe_knn,
                      param_grid=KNNparam_grid,
                      scoring='f1',
                      cv=5,verbose=3,n_jobs=-1)

gs_knn = gs_knn.fit(X_train, y_train)
print(gs_knn.best_score_)
```

```
print(gs_knn.best_score_)
print(gs_knn.best_params_)
```

```
Fitting 5 folds for each of 27 candidates, totalling 135 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:   19.7s
[Parallel(n_jobs=-1)]: Done 124 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 135 out of 135 | elapsed:  1.8min finished
0.43747492002069716
{'kneighborsclassifier__n_neighbors': 25}
```

```
plot_learning_curve(gs_knn.best_estimator_, 'Test Learning Curve', X_train, y_trair
                        n_jobs=None, train_sizes=np.linspace(.1, 1.0, 10))
```

```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.6/dist-packages/ma
```



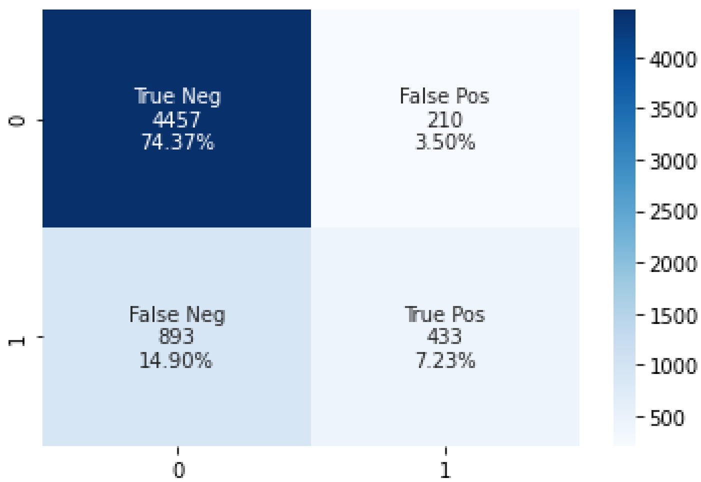

```
clf_knn = gs_knn.best_estimator_ # pick the best model
clf_knn.fit(X_train, y_train)
test_score_knn = '%.3f' % clf_knn.score(X_test, y_test)
print(f'Test accuracy: {test_score_knn}')
```

```
Test accuracy: 0.816
```

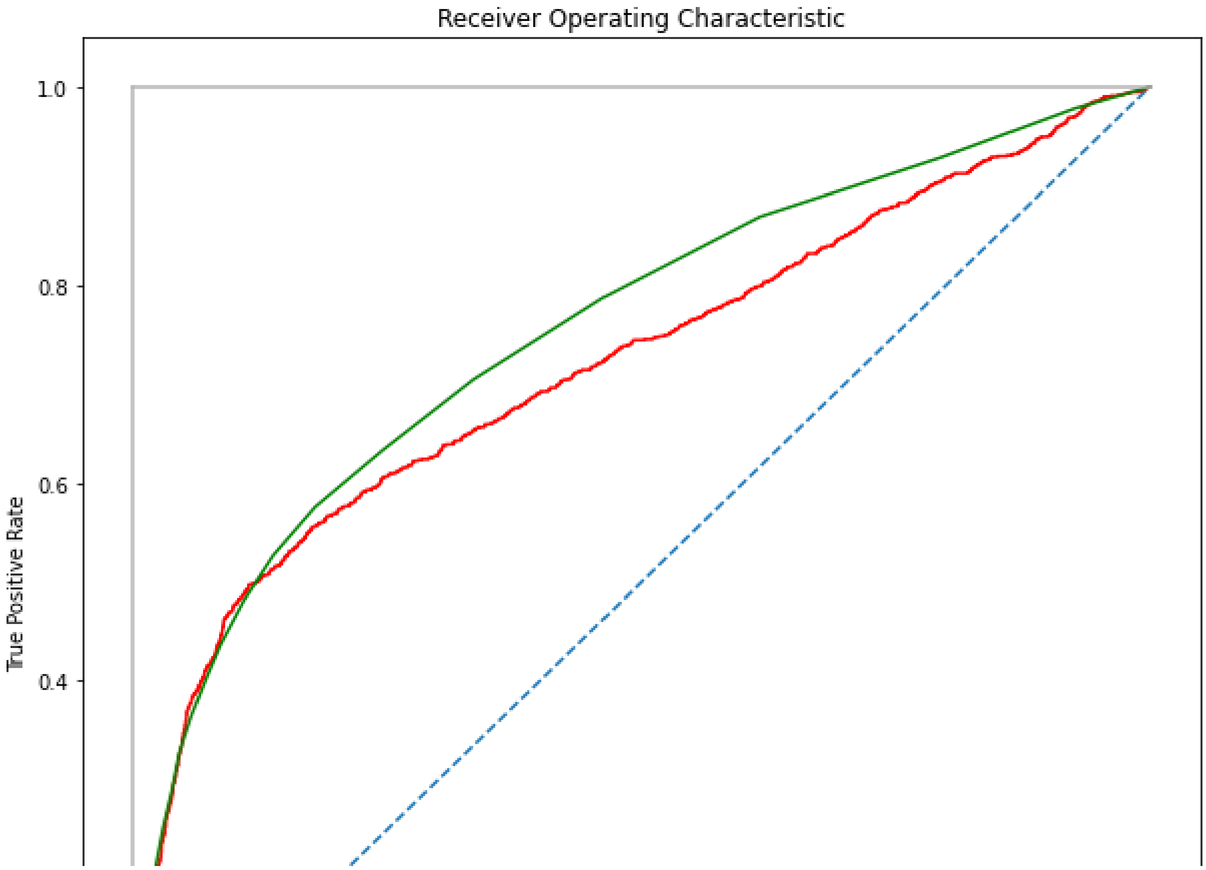
```


y_pred_knn = clf_knn.predict(X_test)
confmat_knn = confusion_matrix(y_true=y_test, y_pred=y_pred_knn)

group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                confmat_knn.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     confmat_knn.flatten()/np.sum(confmat_knn)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
```

```
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(confmat_knn, annot=labels, fmt='', cmap='Blues')

print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred_knn))
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred_knn))
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred_knn))
```

```
Precision: 0.673
Recall: 0.327
F1: 0.440
```




```
y_score_knn = clf_knn.predict_proba(X_test)[:,1]

false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, y_score_lr)
false_positive_rate_knn, true_positive_rate_knn, threshold_knn = roc_curve(y_test,

# Plot ROC curves
plt.subplots(1, figsize=(10,10))
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, color='red',label='Logistic Regre
plt.plot(false_positive_rate_knn, true_positive_rate_knn,color='green', label='KNN'
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc='lower right')
plt.show()
```



DecisionTree Analysis

```
from sklearn.tree import DecisionTreeClassifier

pipe_dt = make_pipeline(RobustScaler(),
                        PCA(n_components=15),
                        DecisionTreeClassifier(criterion='gini', max_depth=4, rando

pipe_dt.fit(X_train, y_train)
y_pred = pipe_dt.predict(X_test)
print('Test Accuracy: %.3f' % pipe_dt.score(X_test, y_test))
```

```
Test Accuracy: 0.807
```

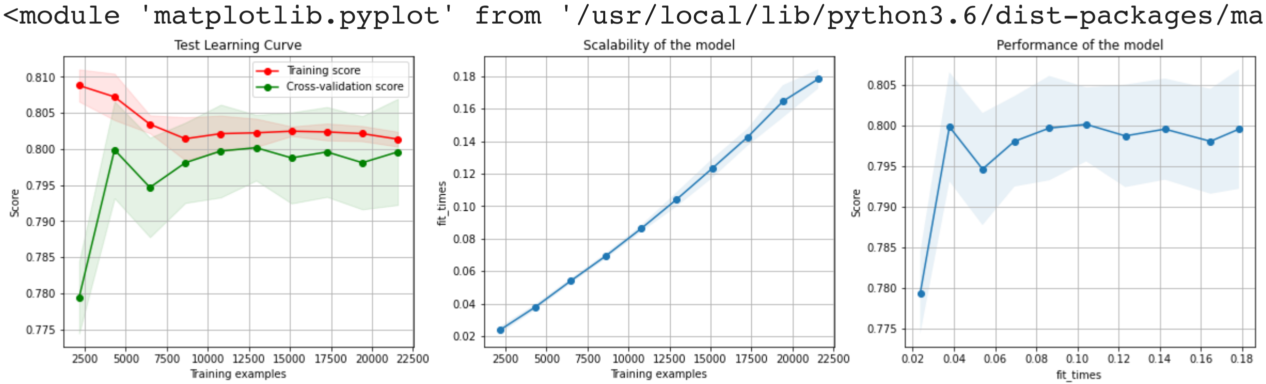
```
DTparam_grid = {
    'decisiontreeclassifier__criterion': ['gini','entropy'],
    'decisiontreeclassifier__max_features': ['auto', 'sqrt', 'log2'],
    'decisiontreeclassifier__min_samples_split': range(2,10),
    'decisiontreeclassifier__min_samples_leaf': range(1,10)
}

gs_dt = GridSearchCV(estimator=pipe_dt,
                     param_grid=DTparam_grid,
                     scoring='f1',
                     cv=10,verbose=3,n_jobs=-1)
```

```

gs_dt = gs_dt.fit(X_train, y_train)
print(gs_dt.best_score_)
print(gs_dt.best_params_)
```

```
Fitting 10 folds for each of 432 candidates, totalling 4320 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    3.5s
[Parallel(n_jobs=-1)]: Done 124 tasks      | elapsed:   15.4s
[Parallel(n_jobs=-1)]: Done 284 tasks      | elapsed:   34.4s
[Parallel(n_jobs=-1)]: Done 508 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 796 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 1148 tasks      | elapsed:  2.3min
[Parallel(n_jobs=-1)]: Done 1564 tasks      | elapsed:  3.2min
[Parallel(n_jobs=-1)]: Done 2044 tasks      | elapsed:  4.1min
[Parallel(n_jobs=-1)]: Done 2588 tasks      | elapsed:  5.4min
[Parallel(n_jobs=-1)]: Done 3196 tasks      | elapsed:  6.8min
[Parallel(n_jobs=-1)]: Done 3868 tasks      | elapsed:  8.4min
[Parallel(n_jobs=-1)]: Done 4320 out of 4320 | elapsed:  9.4min finished
0.36715988747340655
{'decisiontreeclassifier__criterion': 'gini', 'decisiontreeclassifier__max_featu
```

```
plot_learning_curve(gs_dt.best_estimator_, 'Test Learning Curve', X_train, y_train,
                        n_jobs=None, train_sizes=np.linspace(.1, 1.0, 10))
```

```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.6/dist-packages/ma
```

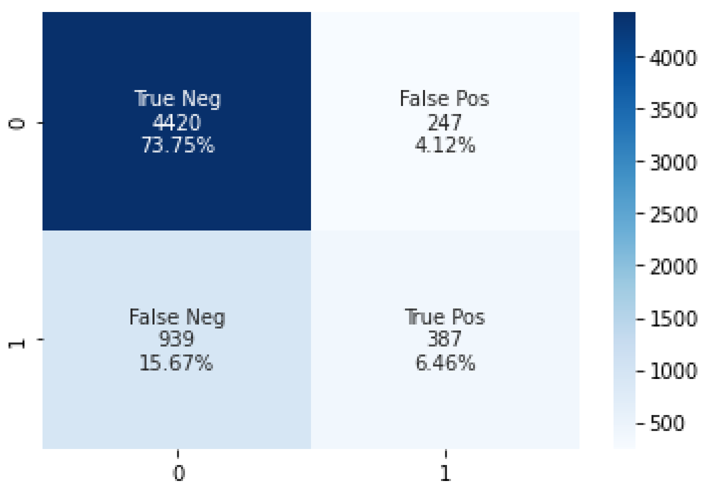



```
clf_dt = gs_dt.best_estimator_ # pick the best model
clf_gra = clf_dt.fit(X_train, y_train)
test_score_dt = '%.3f' % clf_dt.score(X_test, y_test)
print(f'Test accuracy: {test_score_dt}')
```

```
Test accuracy: 0.802
```

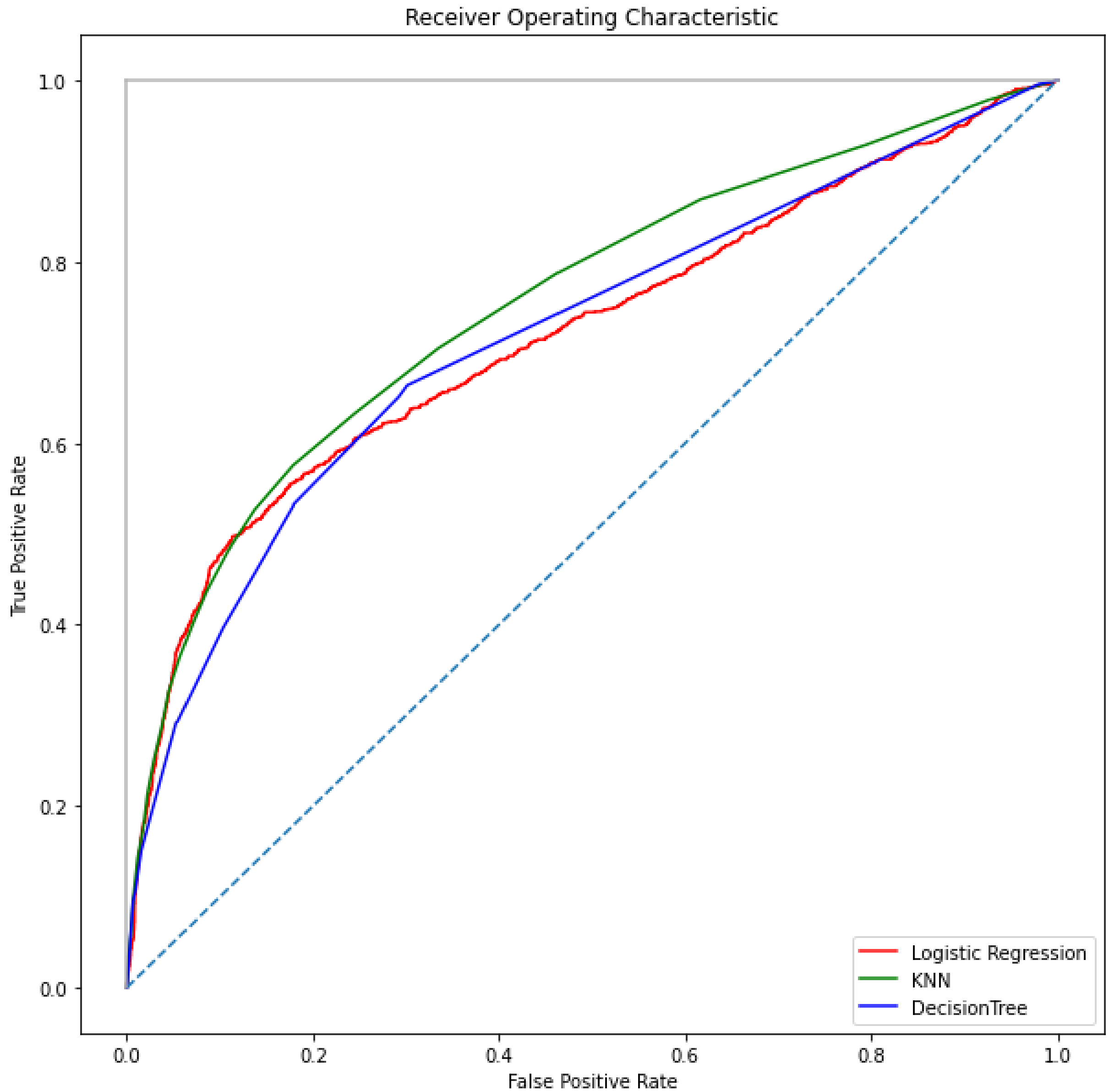
```
y_pred_dt = clf_dt.predict(X_test)
confmat_dt = confusion_matrix(y_true=y_test, y_pred=y_pred_dt)

group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                confmat_dt.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     confmat_dt.flatten()/np.sum(confmat_dt)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(confmat_dt, annot=labels, fmt='', cmap='Blues')

print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred_dt))
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred_dt))
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred_dt))
```

```
Precision: 0.610
Recall: 0.292
F1: 0.395
```




```
y_score_dt = clf_dt.predict_proba(X_test)[:,1]

false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, y_score_lr)
false_positive_rate_knn, true_positive_rate_knn, threshold_knn = roc_curve(y_test,
false_positive_rate_dt, true_positive_rate_dt, threshold_dt = roc_curve(y_test, y_s

# Plot ROC curves
plt.subplots(1, figsize=(10,10))
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, color='red',label='Logistic Regre
plt.plot(false_positive_rate_knn, true_positive_rate_knn,color='green', label='KNN'
plt.plot(false_positive_rate_dt, true_positive_rate_dt,color='blue', label='Decisio
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc='lower right')
```

```
plt.show()
```



RandomForest Analysis

```
from sklearn.ensemble import RandomForestClassifier

pipe_rf = make_pipeline(RobustScaler(),
                        PCA(n_components=15),
                        RandomForestClassifier(criterion='gini',
                                random_state=1,
                                n_jobs=-1) )

pipe_rf.fit(X_train, y_train)
y_pred = pipe_rf.predict(X_test)
print('Test Accuracy: %.3f' % pipe_rf.score(X_test, y_test))
```

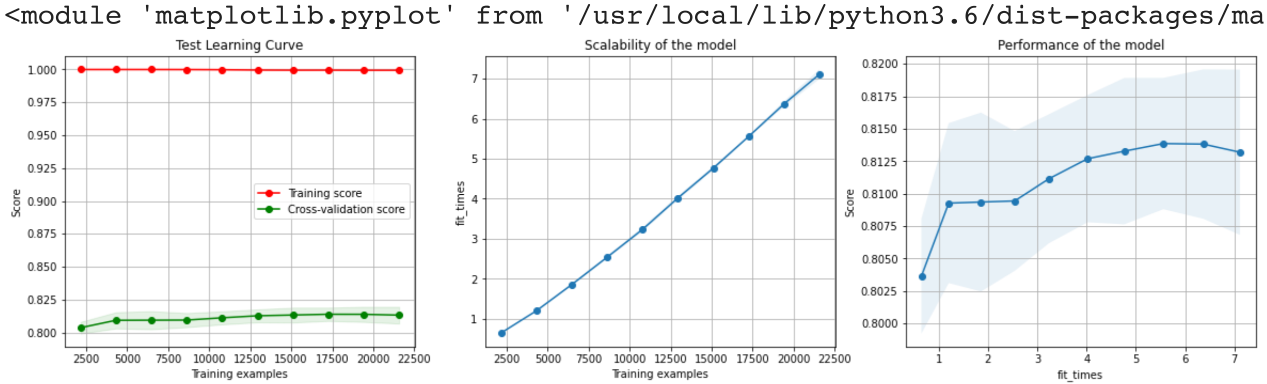
```
Test Accuracy: 0.814
```

```python
RFparam_grid = {
    'randomforestclassifier__criterion': ['gini','entropy']
}

gs_rf = GridSearchCV(estimator=pipe_rf,
                     param_grid=RFparam_grid,
                     scoring='f1',
                     cv=10,verbose=3,n_jobs=-1)

gs_rf = gs_rf.fit(X_train, y_train)
print(gs_rf.best_score_)
print(gs_rf.best_params_)
```

```
Fitting 10 folds for each of 2 candidates, totalling 20 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:  3.1min finished
0.4504975174799739
{'randomforestclassifier__criterion': 'gini'}
```

```python
plot_learning_curve(gs_rf.best_estimator_, 'Test Learning Curve', X_train, y_train,
                        n_jobs=None, train_sizes=np.linspace(.1, 1.0, 10))
```

```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.6/dist-packages/ma
```

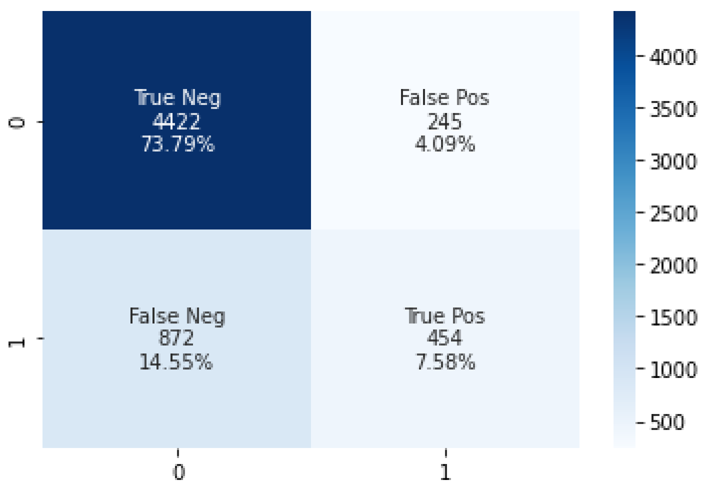



```python
clf_rf = gs_rf.best_estimator_ # pick the best model
clf_rf.fit(X_train, y_train)
test_score_rf = '%.3f' % clf_rf.score(X_test, y_test)
print(f'Test accuracy: {test_score_rf}')
```

```
Test accuracy: 0.814
```

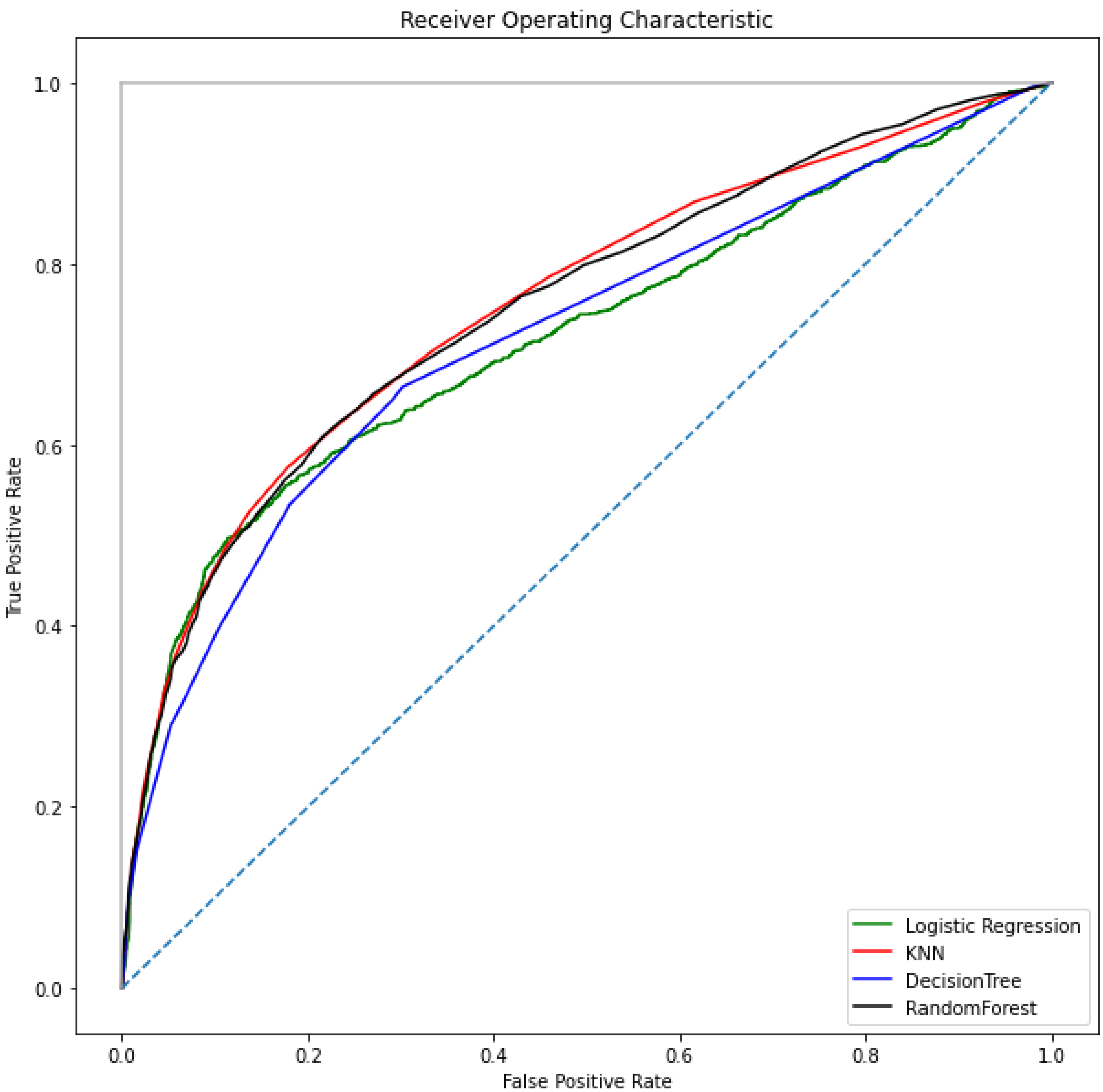
```
y_pred_rf = clf_rf.predict(X_test)
confmat_rf = confusion_matrix(y_true=y_test, y_pred=y_pred_rf)

group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                confmat_rf.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     confmat_rf.flatten()/np.sum(confmat_rf)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(confmat_rf, annot=labels, fmt='', cmap='Blues')

print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred_rf))
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred_rf))
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred_rf))
```

```
Precision: 0.649
Recall: 0.342
F1: 0.448
```




```
y_score_rf = clf_rf.predict_proba(X_test)[:,1]

false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, y_score_lr)
false_positive_rate_knn, true_positive_rate_knn, threshold_knn = roc_curve(y_test,
false_positive_rate_dt, true_positive_rate_dt, threshold_dt = roc_curve(y_test, y_s
false_positive_rate_rf, true_positive_rate_rf, threshold_rf = roc_curve(y_test, y_s

# Plot ROC curves
plt.subplots(1, figsize=(10,10))
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, color='green',label='Logistic Reg
plt.plot(false_positive_rate_knn, true_positive_rate_knn,color='red', label='KNN')
plt.plot(false_positive_rate_dt, true_positive_rate_dt,color='blue', label='Decisic
plt.plot(false_positive_rate_rf, true_positive_rate_rf,color='black', label='Randon
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
```

```
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc='lower right')
plt.show()
```



Results and discussion

```
index= ['LogisticRegression','KNeighbors','DecisionTree','RandomForest']
cols = ['False positives','False Negatives', 'Test Accuracy', 'precision', 'recall'

result = [[confmat[0][1],confmat[1][0],test_score_lr,'%.3f'%precision_score(y_true=
          [confmat_knn[0][1],confmat_knn[1][0],test_score_knn,'%.3f'%precision_scor
          [confmat_dt[0][1],confmat_dt[1][0],test_score_dt,'%.3f'%precision_score(y
          [confmat_rf[0][1],confmat_rf[1][0],test_score_rf,'%.3f'%precision_score(y

df = pd.DataFrame(result, index=index, columns=cols)
```

```
10 df.style.background_gradient(cmap='Blues')
```

	False positives	False Negatives	Test Accuracy	precision	recall	F1score
LogisticRegression	145	998	0.809	0.693	0.247	0.365
KNeighbors	210	893	0.816	0.673	0.327	0.440
DecisionTree	247	939	0.802	0.610	0.292	0.395
RandomForest	245	872	0.814	0.649	0.342	0.448

In Lending, we need to consider false negatives (Good credit clients get rejected) are more acceptable than false positives (bad credit clients get a loan) so we need to choose the model which will give the low False negatives, other way to say is low Recall or F1Score than Accuracy. so far Logistics Regression will be the best model for this Analysis