cogs118b project (1) (1) (1)

December 17, 2020

0.1 COGS 118b Final Project - Unsupervised vs. Supervised on Classification Problem

[27]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import precision_recall_curve
```

[2]:
```
data = pd.read_csv('cardio_train.csv',sep = ';')
data.head()
```

[2]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	\
0	0	18393	2	168	62.0	110	80	1	1	0	
1	1	20228	1	156	85.0	140	90	3	1	0	
2	2	18857	1	165	64.0	130	70	3	1	0	
3	3	17623	2	169	82.0	150	100	1	1	0	
4	4	17474	1	156	56.0	100	60	1	1	0	

	alco	active	cardio
0	0	1	0
1	0	1	1
2	0	0	1
3	0	1	1
4	0	0	0

[3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 13 columns):
id              70000 non-null int64
age             70000 non-null int64
gender          70000 non-null int64
height          70000 non-null int64
weight          70000 non-null float64
ap_hi           70000 non-null int64
```

```
ap_lo          70000 non-null int64
cholesterol    70000 non-null int64
gluc           70000 non-null int64
smoke          70000 non-null int64
alco           70000 non-null int64
active         70000 non-null int64
cardio         70000 non-null int64
dtypes: float64(1), int64(12)
memory usage: 6.9 MB
```

0.1.1 Edit Data

[4]:
```
# id is unuseful
data = data.drop('id', axis = 1)
```

[5]:
```
# change age column
data['age'] = data['age']/365
```

[6]:
```
#find gender to numbers
h1 = data[data["gender"]==1]["height"].mean()
h2 = data[data["gender"]==2]["height"].mean()
w1 = data[data["gender"]==1]["weight"].mean()
w2 = data[data["gender"]==2]["weight"].mean()
print('height1: '+ str(h1) +' height2: ' + str(h2))
print('weight1: '+ str(w1) +' weight2: ' + str(w2))
```

```
height1: 161.35561168460356 height2: 169.94789538210054
weight1: 72.5656050955414 weight2: 77.257306906416
```

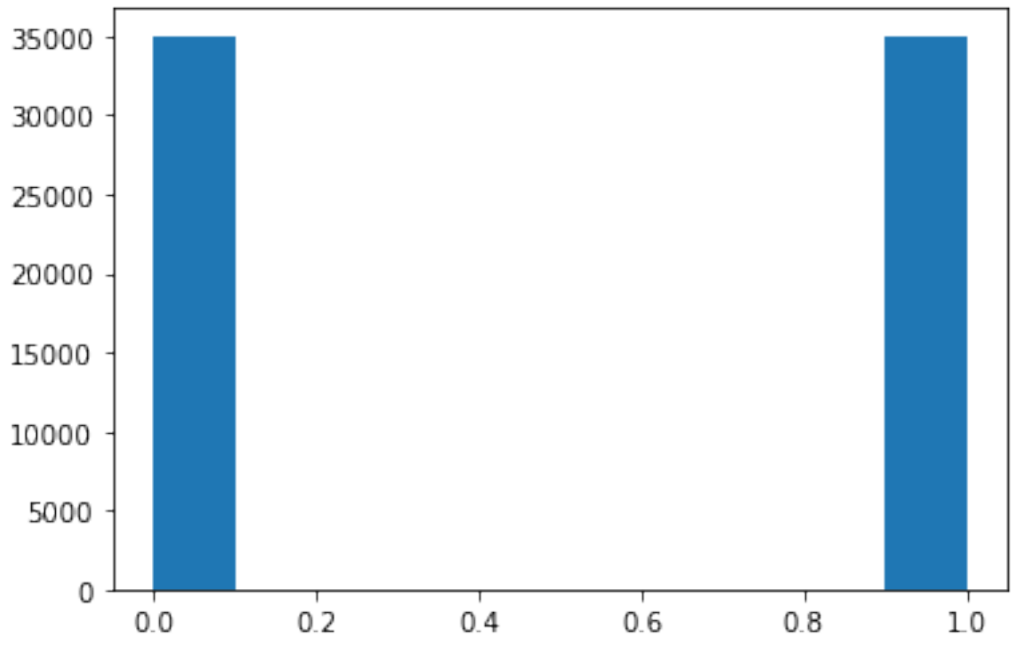
gender == 1 is woman and gender == 2 is man, because the average height and weight of man are higher than woman

[7]:
```
#Now, 1 stands for woman, 0 is man
data["gender"] = data["gender"].apply(lambda x: 0 if x==2 else 1)
```

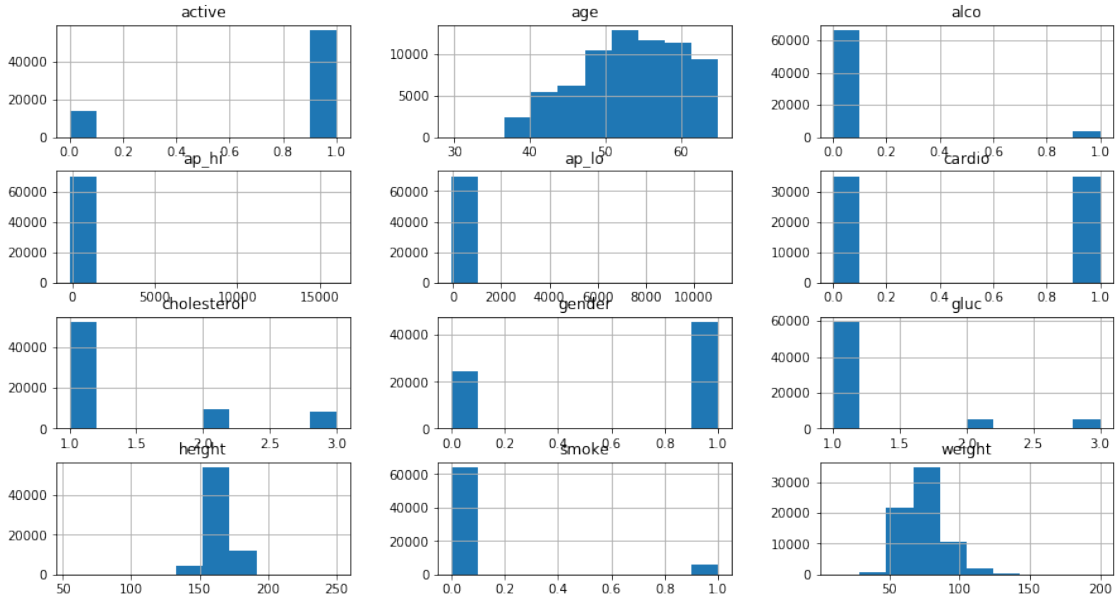
0.2 EDA

[8]:
```
#porpotion of cardiovacular
plt.hist(data['cardio'])
```

[8]:
```
(array([35021.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
            0., 34979.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <a list of 10 Patch objects>)
```



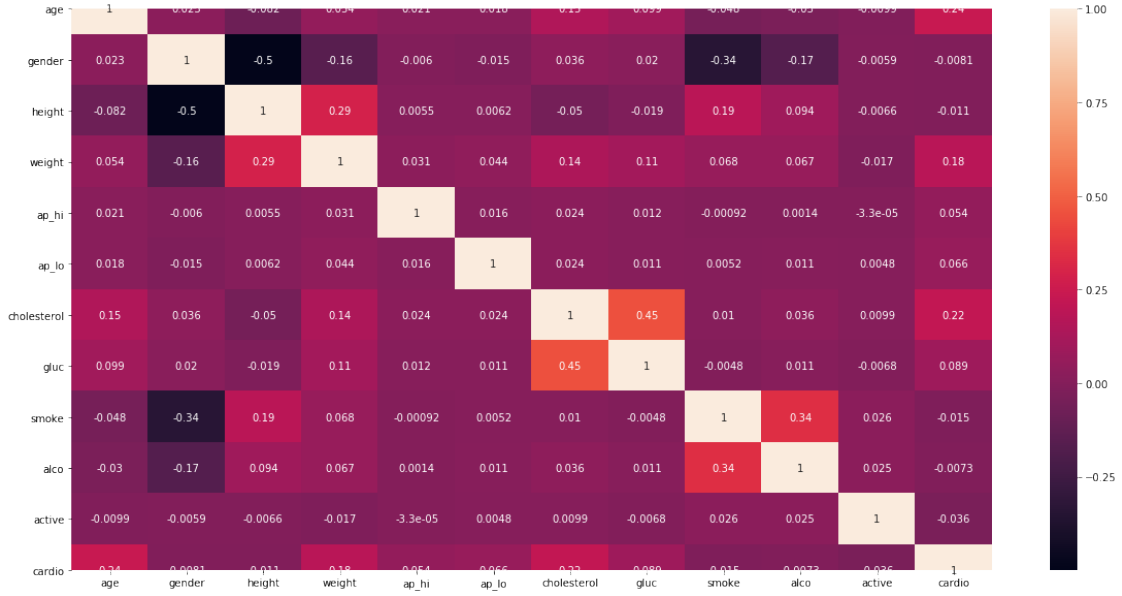
[9]:
```
data.hist(figsize = (15,8))
plt.show()
```



[10]:
```
#heat map
corr = data.corr()
f, ax = plt.subplots(figsize = (20,10))
```

```
sns.heatmap(corr, annot=True, ax=ax)
```

[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff5fb89fac8>



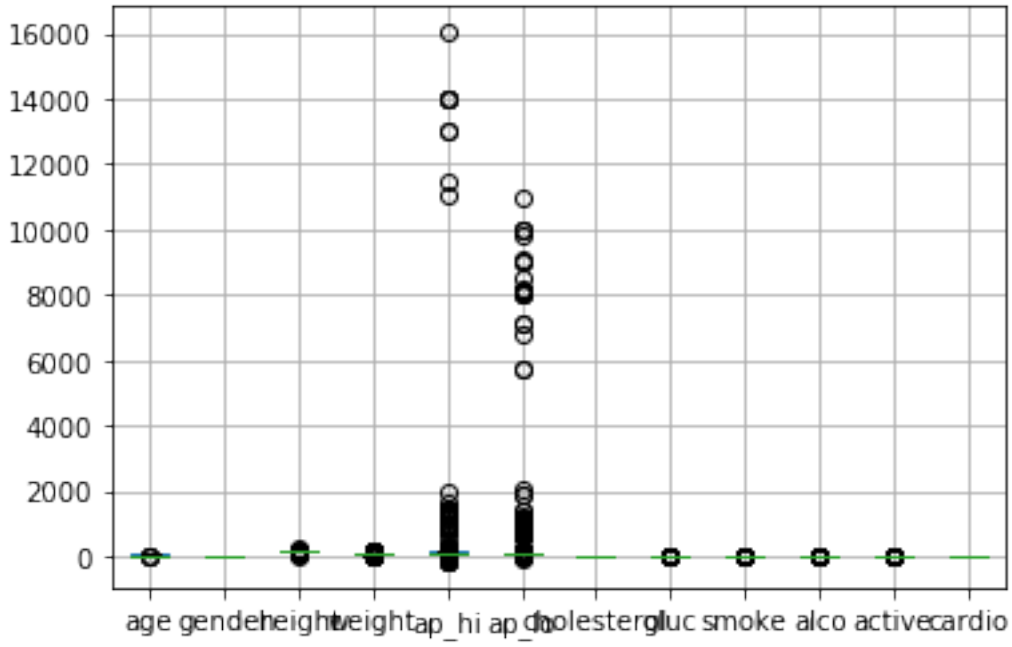
[11]:
```
data.head()
```

[11]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	50.391781	0	168	62.0	110	80	1	1	0	0	1	0
1	55.419178	1	156	85.0	140	90	3	1	0	0	1	1
2	51.663014	1	165	64.0	130	70	3	1	0	0	0	1
3	48.282192	0	169	82.0	150	100	1	1	0	0	1	1
4	47.873973	1	156	56.0	100	60	1	1	0	0	0	0

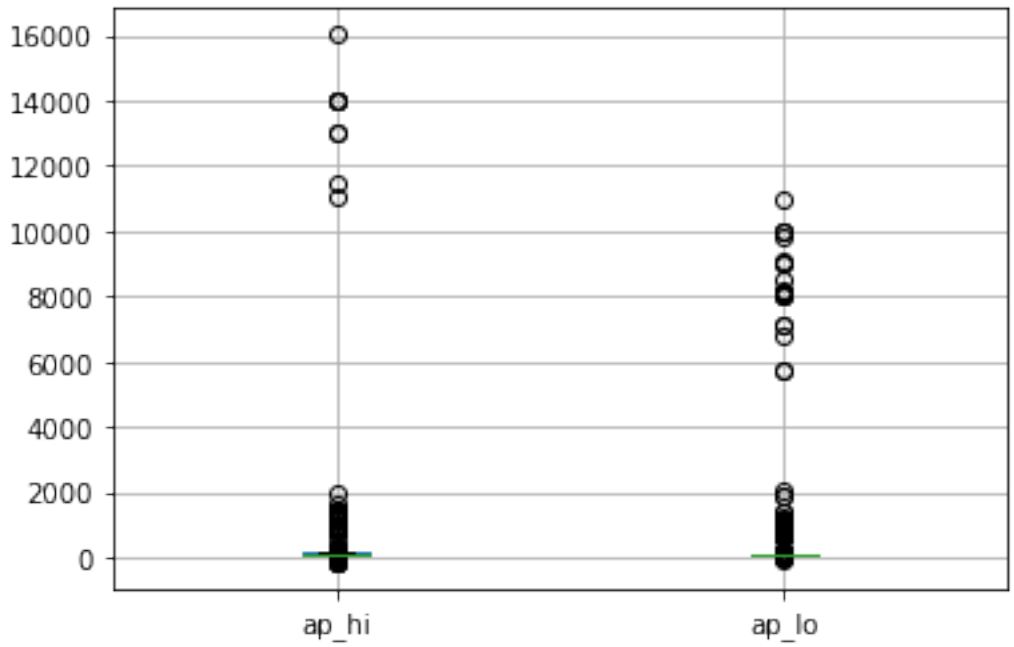
[12]:
```
#check non-binary column
data.boxplot()
```

[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff5f8680278>



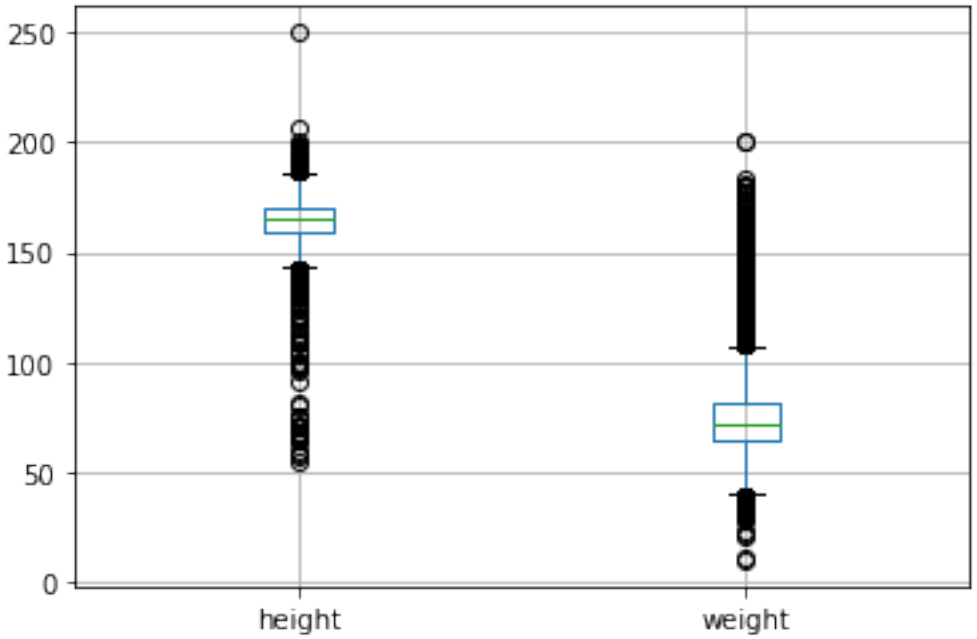
[13]:
```
data.boxplot(column = ['ap_hi', 'ap_lo'])
```

[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff5fb834128>



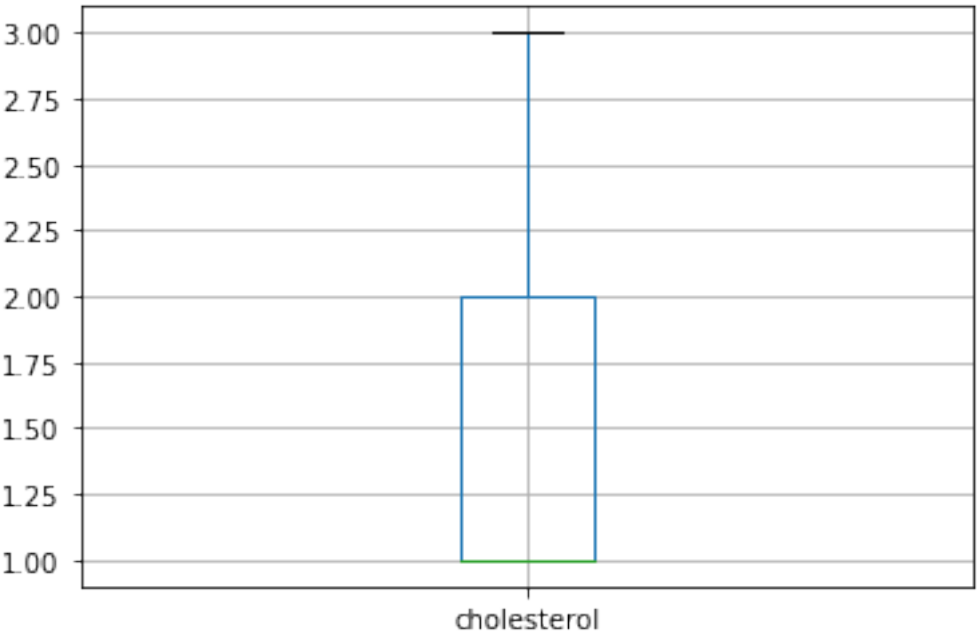
[14]:
```
data.boxplot(column = ['height', 'weight'])
```

[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff5fbaca390>



[15]:
```
data.boxplot(column = ['cholesterol'])
```

[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff5fbdbada0>



[16]:
```
# non_binary = ['height', 'weight','ap_hi', 'ap_lo','cholesterol']
# fig1, ax1 = plt.subplots()
# ax1.set_title('Basic Plot')
# ax1.boxplot(data[non_binary])
```

0.2.1 Train test data

[17]:
```
from sklearn.model_selection import train_test_split
```

[18]:
```
#define x and y
X = data[['age', 'gender', 'height', 'weight', 'ap_hi', 'ap_lo',␣
 ↪'cholesterol','gluc', 'smoke', 'alco', 'active']]
y = data[['cardio']]
```

[19]:
```
#test size 0.2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

0.2.2 Supervised Method - Classification - Logistics Regression

[20]:
```
sacc = []
```

[21]:
```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0).fit(X_train, y_train)
y_predict = clf.predict(X_test)
clf.score(X_test, y_test)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py:724:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

[21]:
```
0.7234285714285714
```

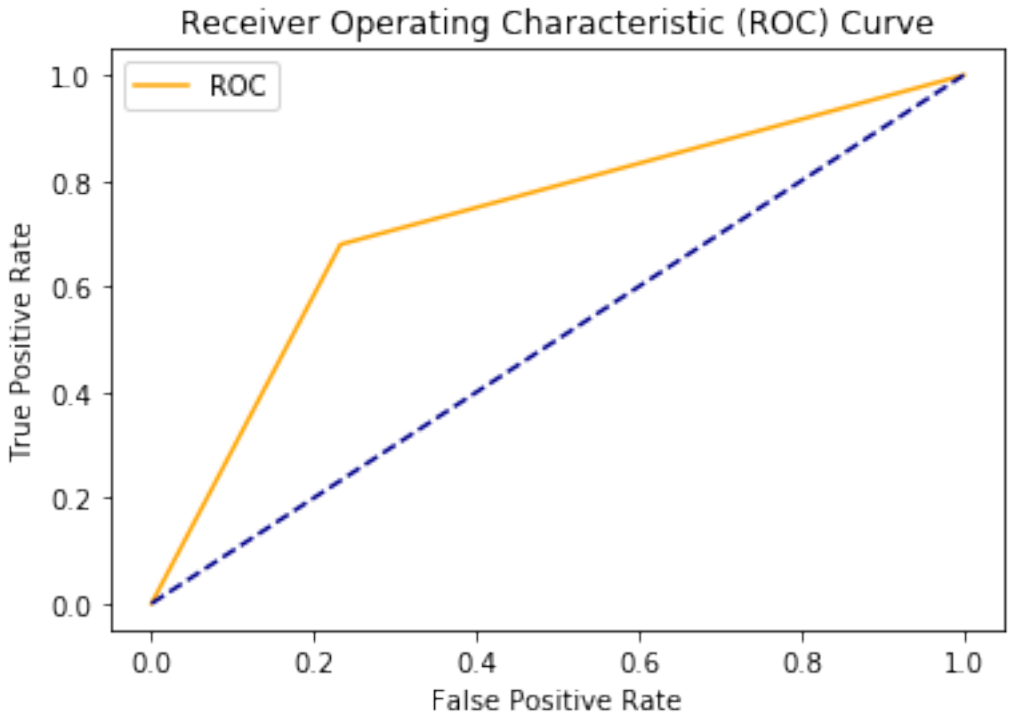
[22]:
```
sacc.append(clf.score(X_test, y_test))
sacc
```

[22]: [0.7234285714285714]

[23]:
```
from sklearn import datasets, metrics, model_selection, svm
from sklearn.metrics import roc_curve
```

[24]:
```
def plot_roc_curve(fper, tper):
    plt.plot(fper, tper, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()
```

[25]:
```
fper_lr, tper_lr, thresholds = roc_curve(y_test, y_predict)
plot_roc_curve(fper_lr, tper_lr)
```



[28]:
```
y_score = clf.decision_function(X_test)
prec_lr, recall_lr, _ = precision_recall_curve(y_test, y_score)
```

[29]:
```
from sklearn.metrics import confusion_matrix
tn, fp, fn, tp = confusion_matrix(y_test, y_predict).ravel()
# Precison
```

```
Precision = tp/(tp+fp)
print("The precision of this logistics model is : ",Precision)
# Recall
Recall= tp/(tp+fn)
print("The Recall score of logistics model is : ",Recall)
# F1 Score
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
print("The F1_Score for this dataset is : ",F1_Score)
```

```
The precision of this logistics model is :  0.7456964006259781
The Recall score of logistics model is :  0.6795493439817456
The F1_Score for this dataset is :  0.7110878973287568
```

0.2.3 Unsupervised - K-Means

[30]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score
```

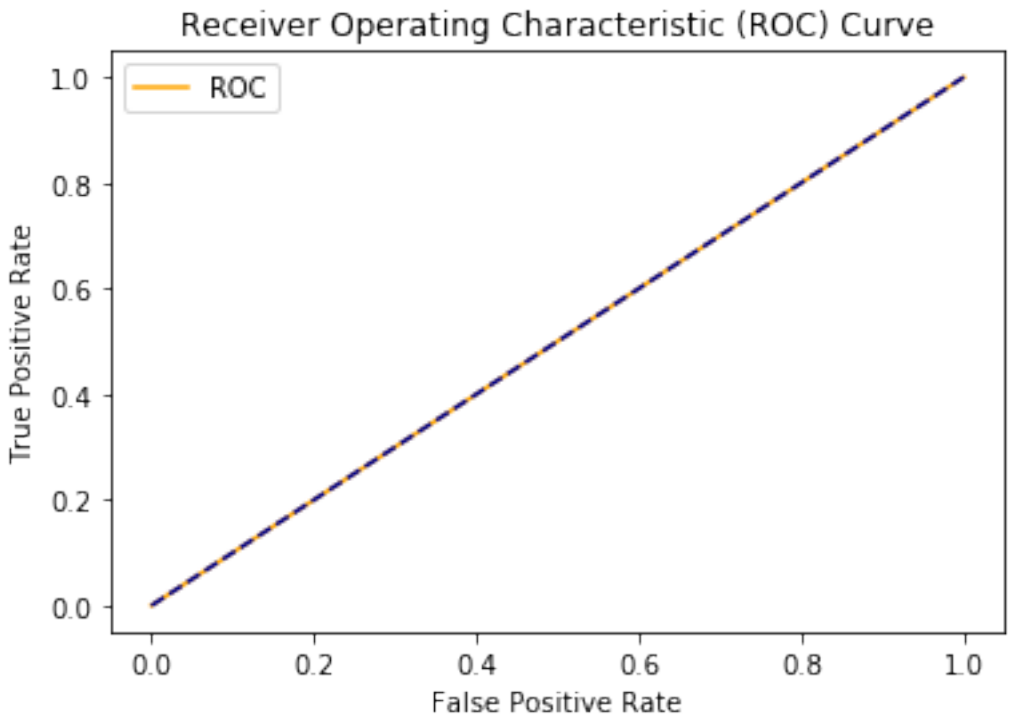
[31]:
```
kmeans = KMeans(n_clusters=2, random_state=0).fit(X_train)
```

[32]:
```
y_predict = kmeans.predict(X_test)
```

[33]:
```
accuracy_score(y_test, y_predict)
```

[33]:
```
0.49914285714285717
```

[34]:
```
fper, tper, thresholds = roc_curve(y_test, y_predict)
plot_roc_curve(fper, tper)
```



[35]:
```
tn, fp, fn, tp = confusion_matrix(y_test, y_predict).ravel()
# Precison
Precision = tp/(tp+fp)
print("The precision of this k-means model is : ",Precision)
# Recall
Recall= tp/(tp+fn)
print("The Recall score of k-means model is : ",Recall)
# F1 Score
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
print("The F1_Score for this dataset is : ",F1_Score)
```

```
The precision of this k-means model is :  0.5
The Recall score of k-means model is :  0.0001426126640045636
The F1_Score for this dataset is :  0.000285143997718848
```

[36]:
```
#Improved
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
X_train, X_validation, y_train, y_validation = train_test_split(X_train,␣
 ↪y_train, test_size=0.2, random_state=42)
```

[37]:
```
# X_train_mod = X_train[['age','weight','cholesterol','gluc','smoke','active']]
# X_test_mod = X_test[['age','weight','cholesterol','gluc','smoke','active']]
```

```
# X_val_mod = X_val[['age','weight','cholesterol','gluc','smoke','active']]
```

[38]:
```
kmeans = KMeans(n_clusters= 6, random_state=0).fit(X_train)
y_predict = kmeans.predict(X_validation)
```

[39]:
```
X_validation['cadio'] = y_validation
X_validation['predict'] = y_predict
df = pd.DataFrame(X_validation.groupby('predict').mean()['cadio'])
df['classification'] = df['cadio'].apply(lambda x: 1 if x>=0.5 else 0)
df['cluster'] = df.index
```

[40]:
```
change = df.to_dict()['classification']
```

[41]:
```
change
```

[41]:
```
{0: 1, 1: 1, 3: 1, 4: 1, 5: 0}
```

[42]:
```
y_predict = kmeans.predict(X_test)
```

[43]:
```
y_result = []
for i in y_predict:
    y_result.append(change[i])
```

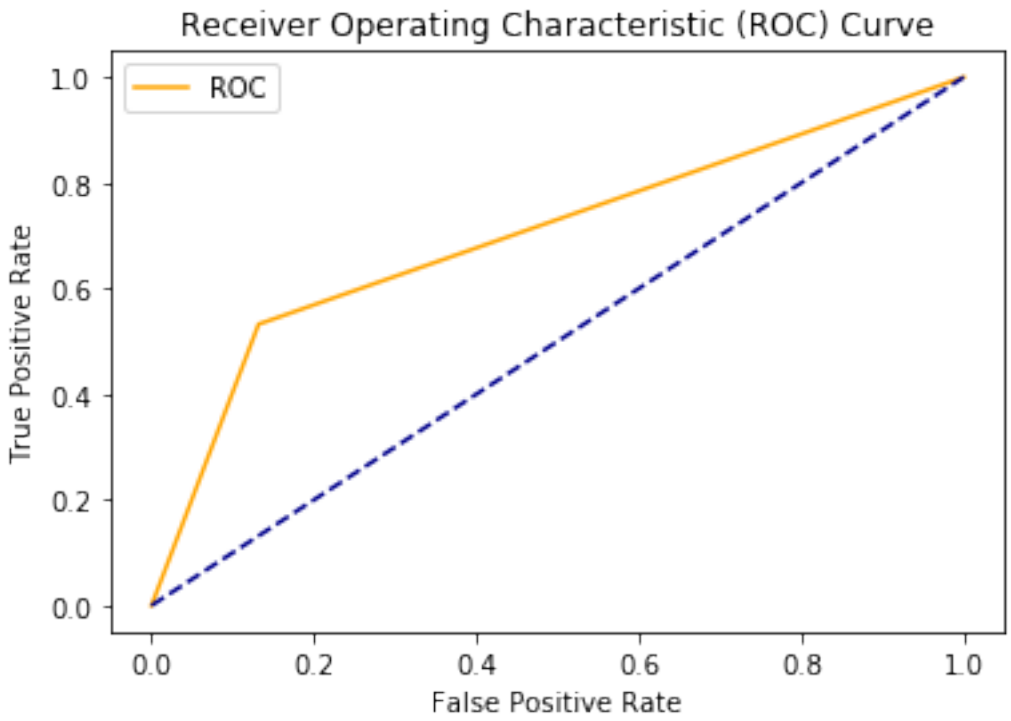
[44]:
```
accuracy_score(y_test, y_result)
```

[44]:
```
0.7
```

[45]:
```
sacc.append(accuracy_score(y_test, y_result))
```

[46]:
```
prec_k, recall_k, _ = precision_recall_curve(y_test, y_result)
```

[47]:
```
fper_k, tper_k, thresholds = roc_curve(y_test, y_result)
plot_roc_curve(fper_k, tper_k)
```



[48]:
```
tn, fp, fn, tp = confusion_matrix(y_test, y_result).ravel()
# Precison
Precision = tp/(tp+fp)
print("The precision of this k-means model is : ",Precision)
# Recall
Recall= tp/(tp+fn)
print("The Recall score of k-means model is : ",Recall)
# F1 Score
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
print("The F1_Score for this dataset is : ",F1_Score)
```

```
The precision of this k-means model is :  0.8021057155135367
The Recall score of k-means model is :  0.5323730747290359
The F1_Score for this dataset is :  0.6399794273958512
```

0.2.4 Supervised - SVM

[49]:
```
from sklearn import svm
clf = svm.SVC(random_state=0).fit(X_train, y_train)
y_predict = clf.predict(X_test)
clf.score(X_test, y_test)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py:724:
```

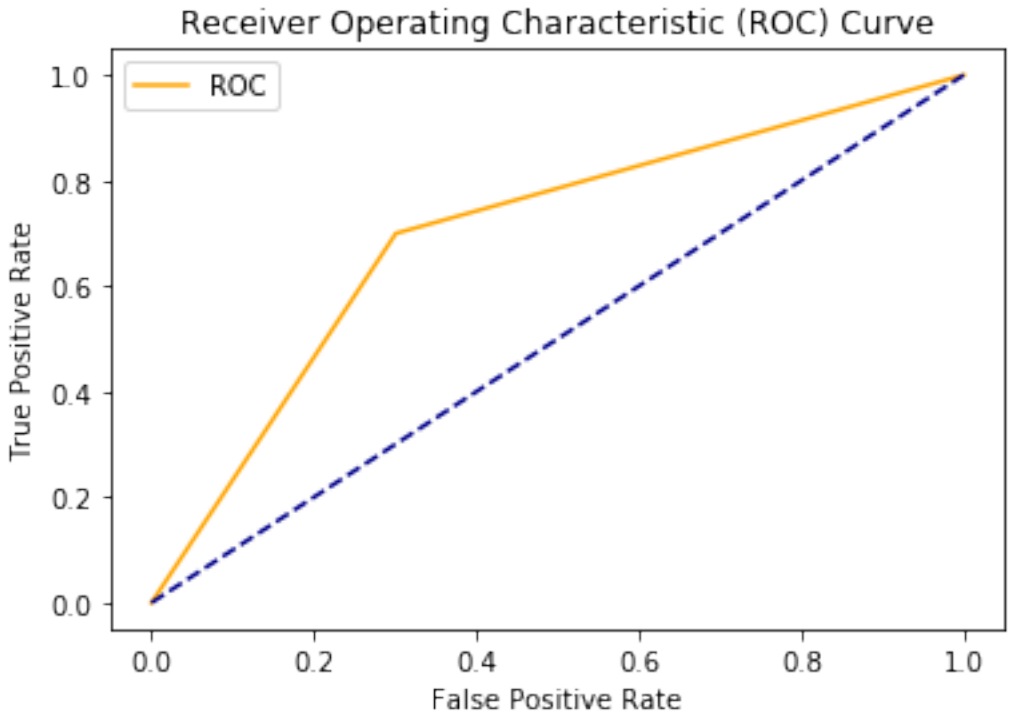
```
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
/opt/conda/lib/python3.7/site-packages/sklearn/svm/base.py:193: FutureWarning:
The default value of gamma will change from 'auto' to 'scale' in version 0.22 to
account better for unscaled features. Set gamma explicitly to 'auto' or 'scale'
to avoid this warning.
  "avoid this warning.", FutureWarning)
```

[49]: 0.6995

[50]:
```
sacc.append(clf.score(X_test, y_test))
```

[51]:
```
y_score = clf.decision_function(X_test)
prec_s, recall_s, _ = precision_recall_curve(y_test, y_score)
```

[52]:
```
fper_s, tper_s, thresholds = roc_curve(y_test, y_predict)
plot_roc_curve(fper_s, tper_s)
```



[53]:
```
tn, fp, fn, tp = confusion_matrix(y_test, y_predict).ravel()
# Precison
Precision = tp/(tp+fp)
print("The precision of SVM model is : ",Precision)
```

```
# Recall
Recall= tp/(tp+fn)
print("The Recall score of SVM model is : ",Recall)
# F1 Score
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
print("The F1_Score for this dataset is : ",F1_Score)
```

```
The precision of SVM model is :  0.7000998715936653
The Recall score of SVM model is :  0.6998003422703936
The F1_Score for this dataset is :  0.6999500748876685
```

0.2.5 Unsupervised - Mixture of Gaussian

[54]:
```
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=2).fit(X_train)
#labels = gmm.predict()
```

[55]:
```
y_predict = gmm.predict(X_test)
```

[56]:
```
accuracy_score(y_test, y_predict)
```

[56]:
```
0.49914285714285717
```

[57]:
```
#Improved
n_com = [3,4,5,6,7,8,9,10]
acc = []
for i in range(len(n_com)):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
    X_train, X_validation, y_train, y_validation = train_test_split(X_train,␣
 ↪y_train, test_size=0.2, random_state = 42)
    gmm = GaussianMixture(n_components=n_com[i], random_state = 42).fit(X_train)
    y_predict = gmm.predict(X_validation)
    X_validation['cadio'] = y_validation
    X_validation['predict'] = y_predict
    df = pd.DataFrame(X_validation.groupby('predict').mean()['cadio'])
    df['classification'] = df['cadio'].apply(lambda x: 1 if x>=0.5 else 0)
    df['cluster'] = df.index
    change = df.to_dict()['classification']
    y_predict = gmm.predict(X_test)
    y_result = []
    for p in y_predict:
        y_result.append(change[p])
    accuracy = accuracy_score(y_test, y_result)
    acc = acc + [accuracy]
```

[58]:
```
acc
```

[58]:
```
[0.5141428571428571,
 0.595,
 0.5008571428571429,
 0.5008571428571429,
 0.5317142857142857,
 0.5311428571428571,
 0.5841428571428572,
 0.5841428571428572]
```

[59]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
X_train, X_validation, y_train, y_validation = train_test_split(X_train,␣
↪y_train, test_size=0.2, random_state=42)
```

[60]:
```
gmm = GaussianMixture(n_components=4, random_state = 42).fit(X_train)
y_predict = gmm.predict(X_validation)
```

[61]:
```
X_validation['cadio'] = y_validation
X_validation['predict'] = y_predict
df = pd.DataFrame(X_validation.groupby('predict').mean()['cadio'])
df['classification'] = df['cadio'].apply(lambda x: 1 if x>=0.5 else 0)
df['cluster'] = df.index
change = df.to_dict()['classification']
change
```

[61]:
```
{0: 0, 2: 1, 3: 1}
```

[62]:
```
y_predict = gmm.predict(X_test)
y_result = []
for i in y_predict:
    y_result.append(change[i])
```

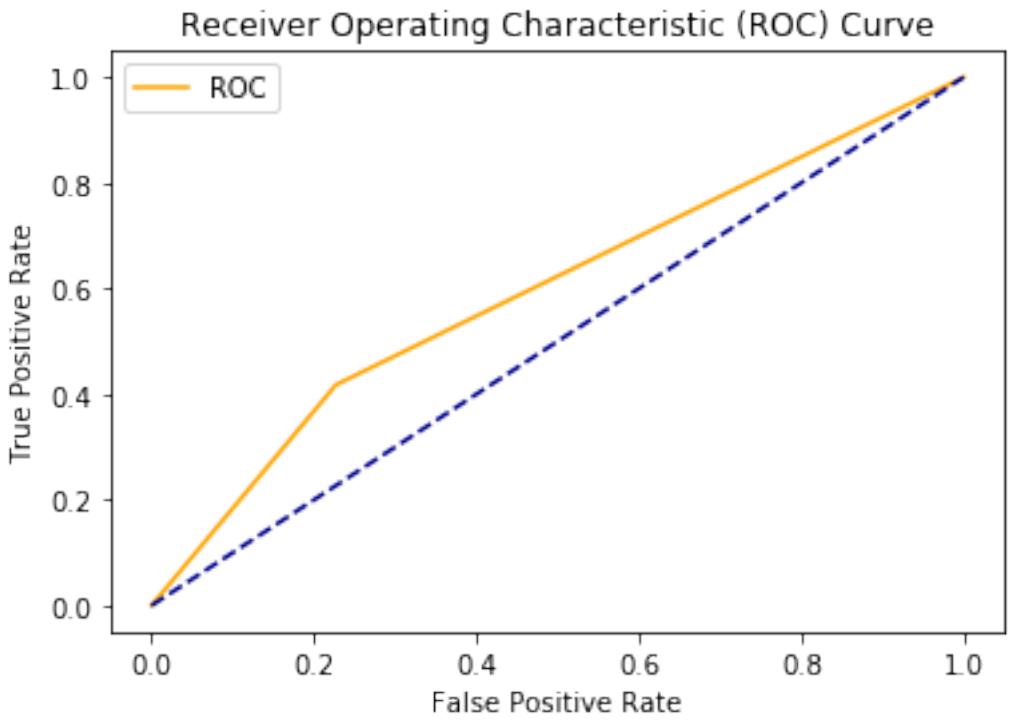
[63]:
```
accuracy_score(y_test, y_result)
```

[63]:
```
0.595
```

[64]:
```
sacc.append(accuracy_score(y_test, y_result))
```

[65]:
```
y_score = gmm.predict_proba(X_test)
y_score = y_score[:,1]
prec_m, recall_m, _ = precision_recall_curve(y_test, y_score)
```

[66]:
```
fper_m, tper_m, thresholds = roc_curve(y_test, y_result)
plot_roc_curve(fper_m, tper_m)
```



[67]:
```
tn, fp, fn, tp = confusion_matrix(y_test, y_result).ravel()
# Precison
Precision = tp/(tp+fp)
print("The precision of this k-means model is : ",Precision)
# Recall
Recall= tp/(tp+fn)
print("The Recall score of k-means model is : ",Recall)
# F1 Score
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
print("The F1_Score for this dataset is : ",F1_Score)
```

```
The precision of this k-means model is :  0.6486486486486487
The Recall score of k-means model is :  0.41756988020536223
The F1_Score for this dataset is :  0.5080687142113484
```

0.2.6 Supervised - Decision tree

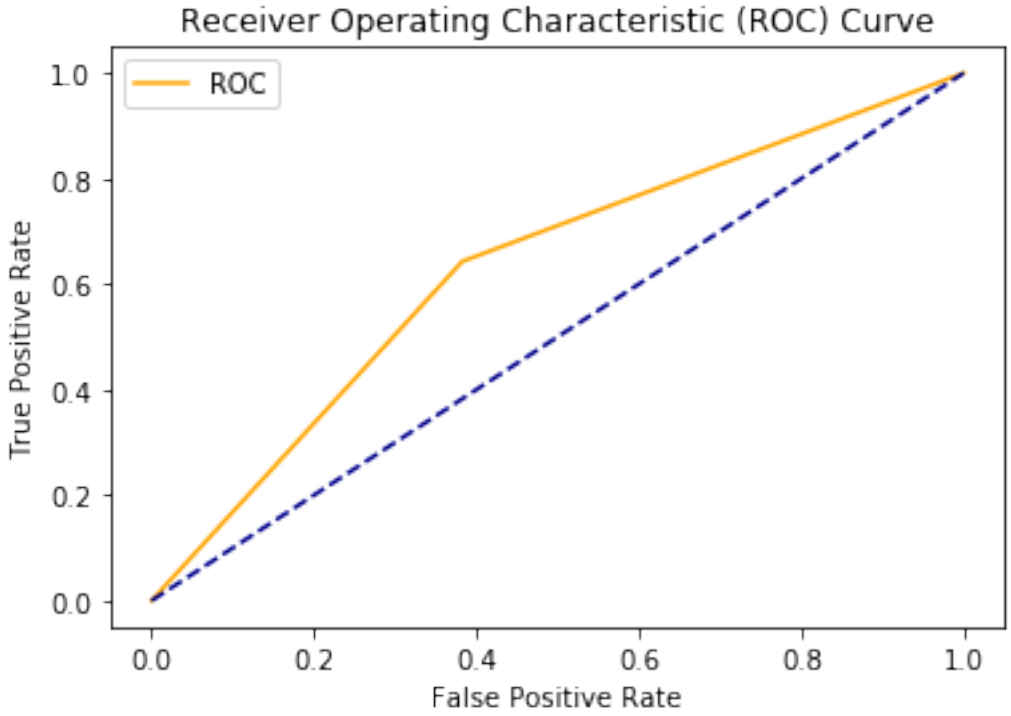
[68]:
```
from sklearn import tree
clf = tree.DecisionTreeClassifier(random_state=0).fit(X_train, y_train)
y_predict = clf.predict(X_test)
clf.score(X_test, y_test)
```

[68]: 0.6302142857142857

[69]:
```
sacc.append(clf.score(X_test, y_test))
```

[70]:
```
y_score = clf.predict_proba(X_test)
y_score = y_score[:,1]
prec_d, recall_d, _ = precision_recall_curve(y_test, y_score)
```

[71]:
```
fper_d, tper_d, thresholds = roc_curve(y_test, y_predict)
plot_roc_curve(fper_d, tper_d)
```



[72]:
```
tn, fp, fn, tp = confusion_matrix(y_test, y_predict).ravel()
# Precison
Precision = tp/(tp+fp)
print("The precision of Decision Tree model is : ",Precision)
# Recall
Recall= tp/(tp+fn)
print("The Recall score of Decision Tree model is : ",Recall)
# F1 Score
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
```

```
The precision of Decision Tree model is :  0.627767720373207
The Recall score of Decision Tree model is :  0.6428978893325727
```

0.2.7 Unsupervised - DBSCAN

[73]:
```
from sklearn.cluster import DBSCAN
```

[74]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[75]:
```
db = DBSCAN(eps=0.3, min_samples=10).fit(X_train)
```

[76]:
```
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)
```

[77]:
```
n_clusters_
```

[77]:
```
13
```

[78]:
```
labels
```

[78]:
```
array([-1, -1, -1, …, -1, -1, -1])
```

[79]:
```
X_train['cadio'] = y_train
X_train['predict'] = labels
df = pd.DataFrame(X_train.groupby('predict').mean()['cadio'])
df['classification'] = df['cadio'].apply(lambda x: 1 if x>=0.5 else 0)
df['cluster'] = df.index
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

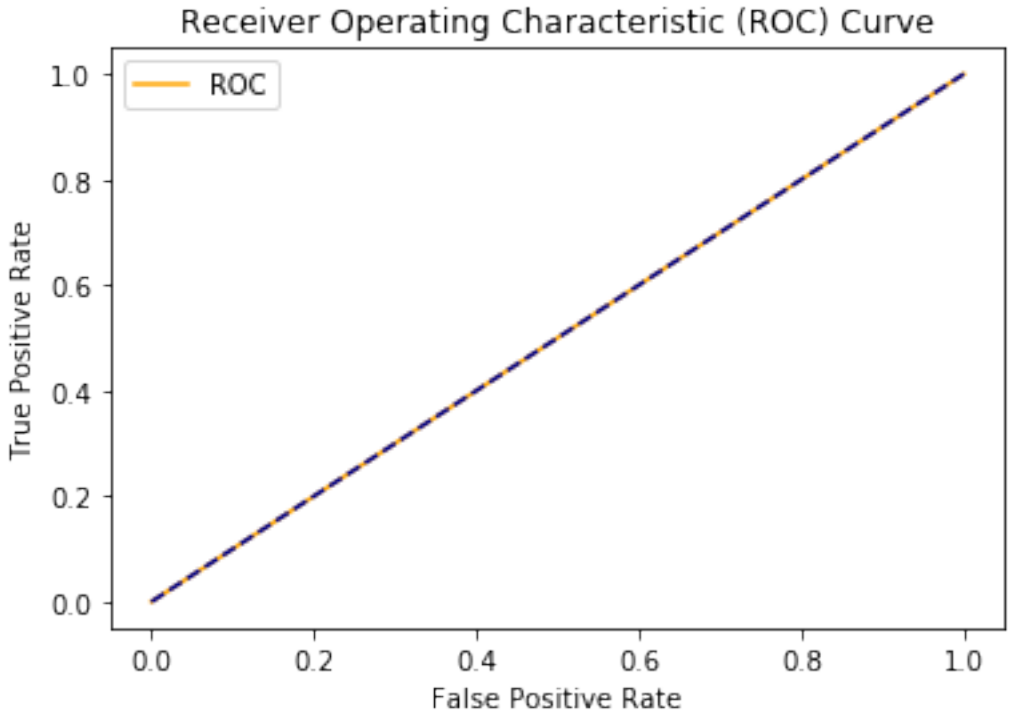
[80]:
```
change = df.to_dict()['classification']
y_predict = DBSCAN(eps=0.3, min_samples=10).fit(X_test).labels_
y_result = []
for p in y_predict:
    y_result.append(change[p])
accuracy = accuracy_score(y_test, y_result)
```

[81]:
```
accuracy
```

[81]: 0.5008571428571429

[82]:
```
sacc.append(accuracy)
```

[83]:
```
fper_db, tper_db, thresholds = roc_curve(y_test, y_result)
plot_roc_curve(fper_db, tper_db)
```



[84]:
```
prec_db, recall_db, _ = precision_recall_curve(y_test, y_result)
```

[86]:
```
eps = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]
acc = []
for i in range(len(eps)):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

```
    db = DBSCAN(eps= eps[i], min_samples=10).fit(X_train)
    labels = db.labels_
    X_train['cadio'] = y_train
    X_train['predict'] = labels
    df = pd.DataFrame(X_train.groupby('predict').mean()['cadio'])
    df['classification'] = df['cadio'].apply(lambda x: 1 if x>=0.5 else 0)
    df['cluster'] = df.index

    change = df.to_dict()['classification']
    y_predict = DBSCAN(eps=0.3, min_samples=10).fit(X_test).labels_
    y_result = []
    for p in y_predict:
        y_result.append(change[p])
    accuracy = accuracy_score(y_test, y_result)
    acc = acc + [accuracy]
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[87]:
```
acc
```

[87]:
```
[0.49914285714285717,
 0.49914285714285717,
 0.5008571428571429,
 0.5008571428571429,
 0.5008571428571429,
 0.5008571428571429,
 0.5008571428571429,
 0.5008571428571429,
 0.5008571428571429,
 0.5008571428571429]
```

[88]:
```
sam = [10, 20, 30, 40, 50, 60, 70, 80, 90 , 100]
acc = []
for i in range(len(eps)):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
    db = DBSCAN(eps= 0.3, min_samples= sam[i]).fit(X_train)
    labels = db.labels_
    X_train['cadio'] = y_train
    X_train['predict'] = labels
    df = pd.DataFrame(X_train.groupby('predict').mean()['cadio'])
    df['classification'] = df['cadio'].apply(lambda x: 1 if x>=0.5 else 0)
    df['cluster'] = df.index

    change = df.to_dict()['classification']
    y_predict = DBSCAN(eps=0.3, min_samples=10).fit(X_test).labels_
    y_result = []
    for p in y_predict:
        y_result.append(change[p])
    accuracy = accuracy_score(y_test, y_result)
    acc = acc + [accuracy]
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[89]:
```
acc
```

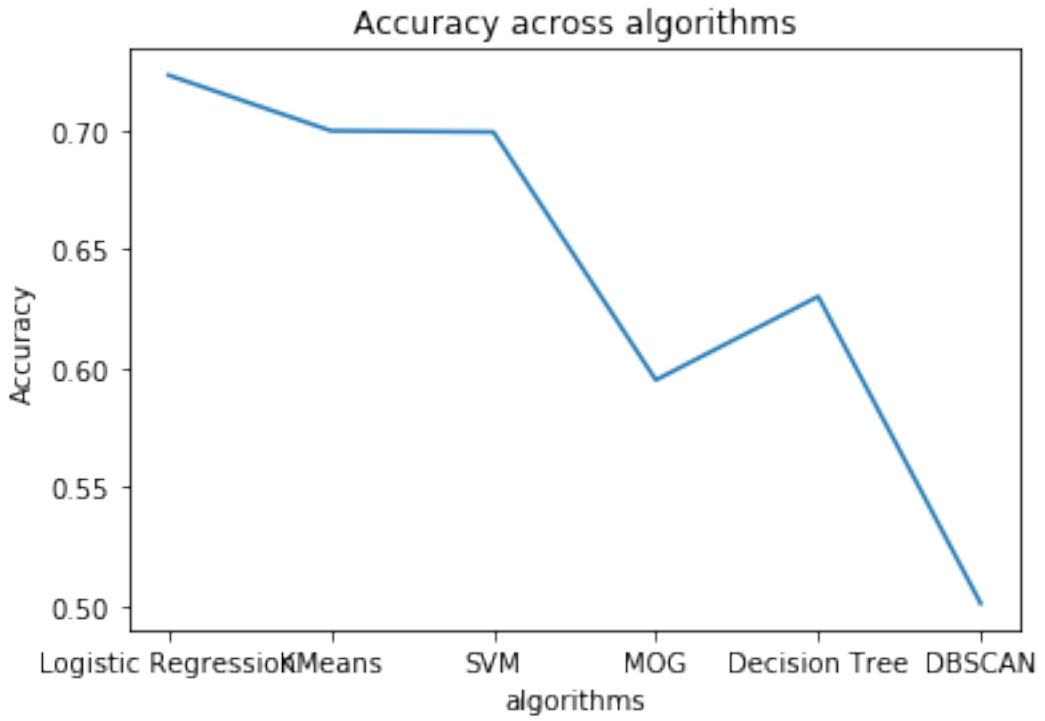
[89]:
```
[0.5008571428571429,
 0.49914285714285717,
 0.49914285714285717,
 0.49914285714285717,
 0.49914285714285717,
 0.49914285714285717,
```

```
0.49914285714285717,
0.49914285714285717,
0.49914285714285717,
0.49914285714285717]
```

[90]:
```
name = ['Logistic Regression','KMeans','SVM', 'MOG','Decision Tree','DBSCAN']
```

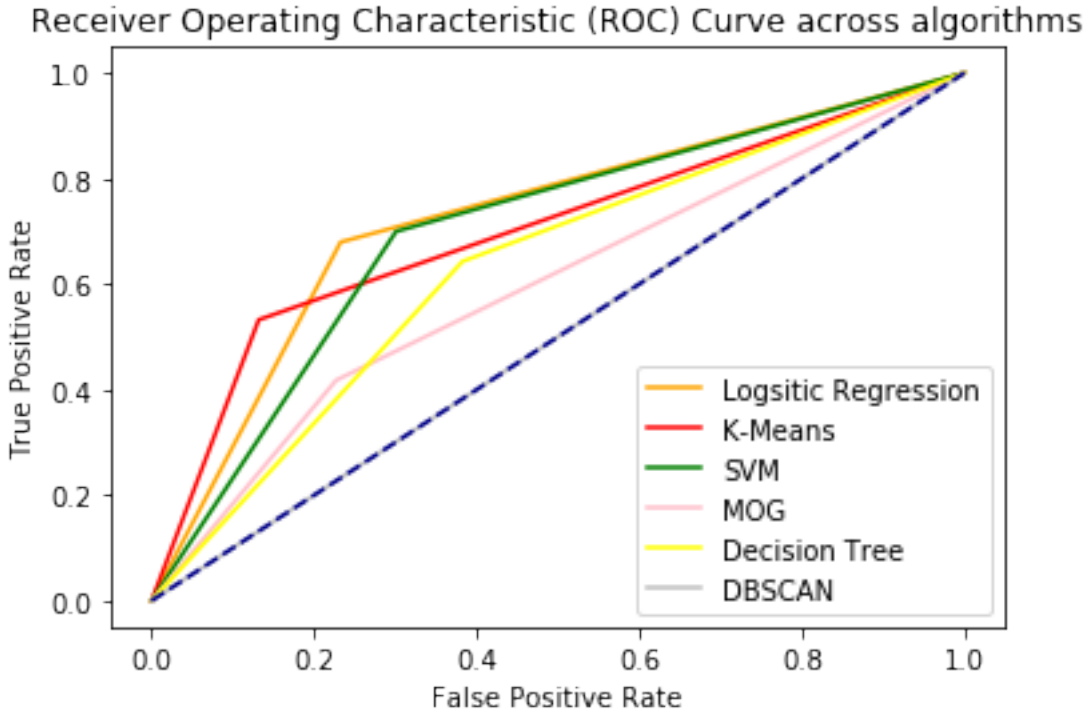
[91]:
```
plt.plot(name,sacc)
plt.xlabel('algorithms')
plt.ylabel('Accuracy')
plt.title('Accuracy across algorithms')
```

[91]:
```
Text(0.5, 1.0, 'Accuracy across algorithms')
```

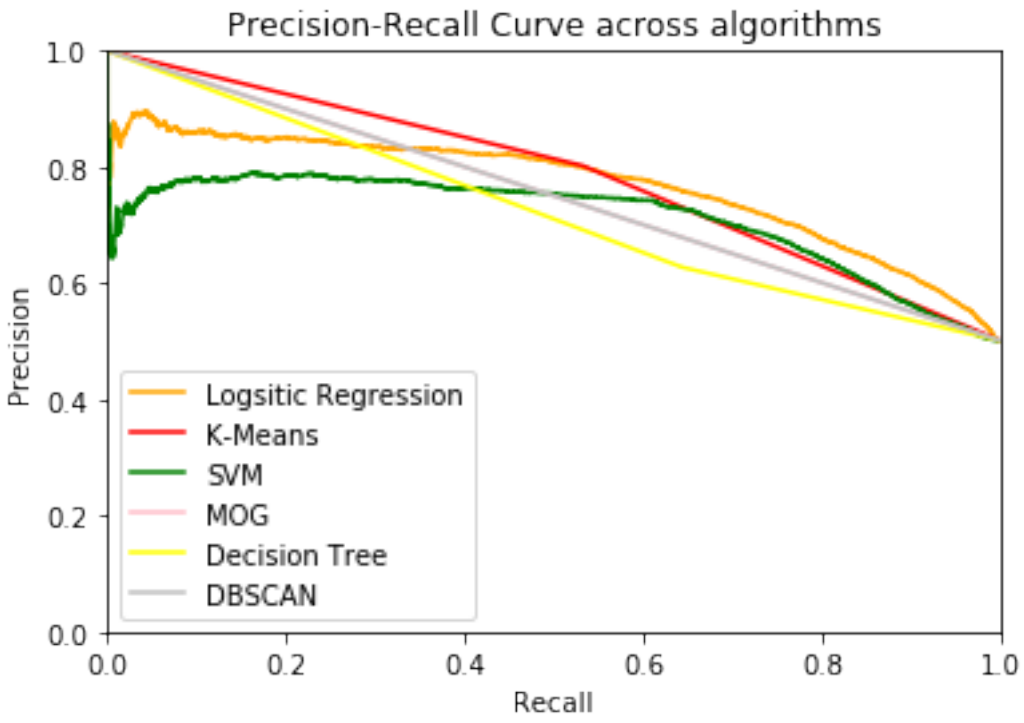


[92]:
```
plt.plot(fper_lr, tper_lr, color='orange', label='Logsitic Regression')
plt.plot(fper_k, tper_k, color='red', label='K-Means')
plt.plot(fper_s, tper_s, color='green', label='SVM')
plt.plot(fper_m, tper_m, color='pink', label='MOG')
plt.plot(fper_d, tper_d, color='yellow', label='Decision Tree')
plt.plot(fper_db, tper_db, color='silver', label='DBSCAN')
plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
plt.title('Receiver Operating Characteristic (ROC) Curve across algorithms')
plt.legend()
plt.show()
```



[93]:
```
plt.plot(recall_lr, prec_lr, color='orange', label='Logsitic Regression')
plt.plot(recall_k, prec_k, color='red', label='K-Means')
plt.plot(recall_s, prec_s, color='green', label='SVM')
plt.plot(recall_m, prec_m, color='pink', label='MOG')
plt.plot(recall_d, prec_d, color='yellow', label='Decision Tree')
plt.plot(recall_db, prec_db, color='silver', label='DBSCAN')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.0])
plt.xlim([0.0, 1.0])
plt.title('Precision-Recall Curve across algorithms')
plt.legend()
plt.show()
```


Precision-Recall Curve across algorithms
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Logsitic Regression
K-Means
SVM
MOG
Decision Tree
DBSCAN

[ ]: