Assignment 3: Classification Models Project

- Part 1: Running classification models we learned using help from Tutorial_6 on your dataset
- Part 2: Feature selection and Data splitting on the provided dataset
- Part 3: Running the classification models on the provided dataset
- Part 4: Performing accuracy analysis and comparing the models on the provided dataset

Team 12 Members:

- Alicia Luna
- Derek Dilger
- Gary Young
- Ian Schultz
- Jesus Cervantes
- Matthew Mendoza
- Thomas Jaramillo-Ochoa
- Yun-jeong Lee

Part 1: Running classification models we learned using help from Tutorial_6 on your dataset

Decision Tree Regressor

In [ ]:
```
import numpy as np
import csv
import matplotlib.pyplot as plt
import pandas as pd

with open('./data/uci_heartDisease_changed.csv', 'r') as f:
    reader = csv.reader(f)
    data = list(reader)

data_array = np.array(data)

X = data_array[1:, 4:5].astype(int)
y = data_array[1:, 13].astype(int)

print(X)
print(y)
```

```
[[233]
 [286]
 [229]
 [250]
 [204]
 [236]
 [268]
 [354]
 [254]
 [203]
 [192]
 [294]
 [256]
 [263]
 [199]
 [168]
 [229]
 [239]
 [275]
 [266]
 [211]
 [283]
 [284]
 [224]
 [206]
 [219]
 [340]
 [226]
 [247]
 [167]
 [239]
 [230]
 [335]
 [234]
 [233]
 [226]
 [177]
 [276]
 [353]
 [243]
 [225]
 [199]
 [302]
 [212]
 [330]
 [230]
 [175]
 [243]
 [417]
 [197]
 [198]
 [177]
 [290]
 [219]
 [253]
 [266]
```

[233]
[172]
[273]
[213]
[305]
[177]
[216]
[304]
[188]
[282]
[185]
[232]
[326]
[231]
[269]
[254]
[267]
[248]
[197]
[360]
[258]
[308]
[245]
[270]
[208]
[264]
[321]
[274]
[325]
[235]
[257]
[216]
[234]
[256]
[302]
[164]
[231]
[141]
[252]
[255]
[239]
[258]
[201]
[222]
[260]
[182]
[303]
[265]
[188]
[309]
[177]
[229]
[260]
[219]
[307]
[249]

[186]
[341]
[263]
[203]
[211]
[183]
[330]
[254]
[256]
[407]
[222]
[217]
[282]
[234]
[288]
[239]
[220]
[209]
[258]
[227]
[204]
[261]
[213]
[250]
[174]
[281]
[198]
[245]
[221]
[288]
[205]
[309]
[240]
[243]
[289]
[250]
[308]
[318]
[298]
[265]
[564]
[289]
[246]
[322]
[299]
[300]
[293]
[277]
[197]
[304]
[214]
[248]
[255]
[207]
[223]
[288]

[282]
[160]
[269]
[226]
[249]
[394]
[212]
[274]
[233]
[184]
[315]
[246]
[274]
[409]
[244]
[270]
[305]
[195]
[240]
[246]
[283]
[254]
[196]
[298]
[247]
[294]
[211]
[299]
[234]
[236]
[244]
[273]
[254]
[325]
[126]
[313]
[211]
[309]
[259]
[200]
[262]
[244]
[215]
[231]
[214]
[228]
[230]
[193]
[204]
[243]
[303]
[271]
[268]
[267]
[199]
[282]

[269]
[210]
[204]
[277]
[206]
[212]
[196]
[327]
[149]
[269]
[201]
[286]
[283]
[249]
[271]
[295]
[235]
[306]
[269]
[234]
[178]
[237]
[234]
[275]
[212]
[208]
[201]
[218]
[263]
[295]
[303]
[209]
[223]
[197]
[245]
[261]
[242]
[319]
[240]
[226]
[166]
[315]
[204]
[218]
[223]
[180]
[207]
[228]
[311]
[149]
[204]
[227]
[278]
[220]
[232]
[197]

```
 [335]
 [253]
 [205]
 [192]
 [203]
 [318]
 [225]
 [220]
 [221]
 [240]
 [212]
 [342]
 [169]
 [187]
 [197]
 [157]
 [176]
 [241]
 [264]
 [193]
 [131]
 [236]
 [175]]
[0 2 1 0 0 0 3 0 2 1 0 0 2 0 0 0 1 0 0 0 0 0 1 3 4 0 0 0 0 3 0 2 1 0 0 0 3
 1 3 0 4 0 0 0 1 4 0 4 0 0 0 0 2 0 1 1 1 1 0 0 2 0 1 0 2 2 1 0 2 1 0 3 1 1
 1 0 1 0 0 3 0 0 0 3 0 0 0 0 0 0 0 3 0 0 0 1 2 3 0 0 0 0 0 0 3 0 2 1 2 3 1
 1 0 2 2 0 0 0 3 2 3 4 0 3 1 0 3 3 0 0 0 0 0 0 0 0 4 3 1 0 0 1 0 1 0 1 4 0
 0 0 0 0 0 4 3 1 1 1 2 0 0 4 0 0 0 0 0 0 1 0 3 0 1 0 4 1 0 1 0 0 3 2 0 0 1
 0 0 2 1 2 0 3 1 2 0 3 0 0 0 1 0 0 0 0 0 3 3 3 0 1 0 4 0 3 1 0 0 0 0 0 0 0
 0 3 1 0 0 0 3 2 0 2 1 0 0 3 2 1 0 0 0 0 0 2 0 2 2 1 3 0 0 1 0 0 0 0 0 0 0
 1 0 3 0 0 4 2 2 2 1 0 1 0 2 0 1 0 0 0 1 0 2 0 3 0 2 4 2 0 0 0 1 0 2 2 1 0
 3 1 1 2 3 1 0]
```

In [ ]:
```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state=0)
regressor.fit(X, y)
```

Out[ ]:
```
DecisionTreeRegressor(random_state=0)
```

In [ ]:
```
y_pred = regressor.predict([[298]])
print("Predicted Heart Disease Diagnosis: % d\n" % y_pred)
```

```
Predicted Heart Disease Diagnosis:  1
```

```
C:\Users\matrn\AppData\Local\Temp\ipykernel_18248\118750979.py:2: DeprecationWarnin
g: Conversion of an array with ndim > 0 to a scalar is deprecated, and will error in
future. Ensure you extract a single element from your array before performing this o
peration. (Deprecated NumPy 1.25.)
  print("Predicted Heart Disease Diagnosis: % d\n" % y_pred)
```

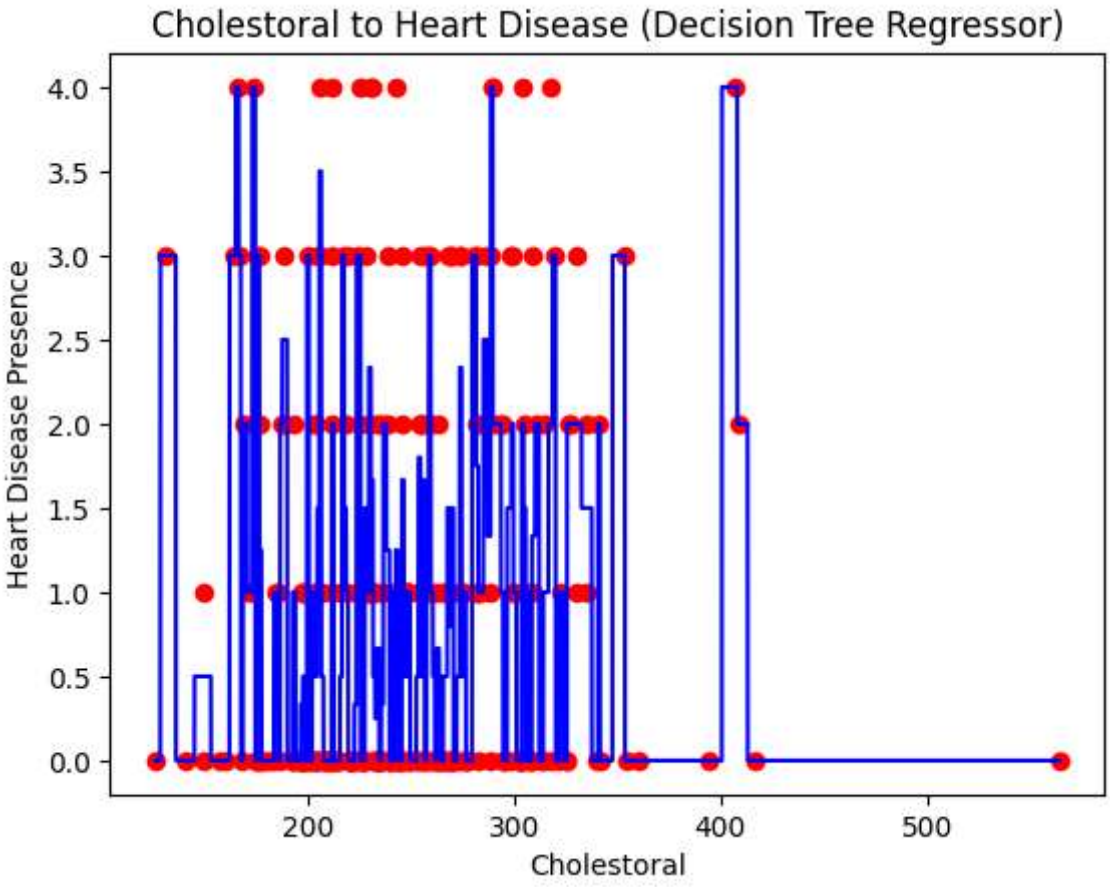
arrange for creating a range of values from min value of X to max value of X with a difference of 0.01 between two consecutive values

In [ ]:
```
X_grid = np.arange(min(X), max(X), 0.01)
```

```
C:\Users\matrn\AppData\Local\Temp\ipykernel_18248\109639871.py:1: DeprecationWarnin
g: Conversion of an array with ndim > 0 to a scalar is deprecated, and will error in
future. Ensure you extract a single element from your array before performing this o
peration. (Deprecated NumPy 1.25.)
  X_grid = np.arange(min(X), max(X), 0.01)
```

reshape for reshaping the data into a `len(X_grid)*1` array, i.e. to make a column out of the `X_grid` values

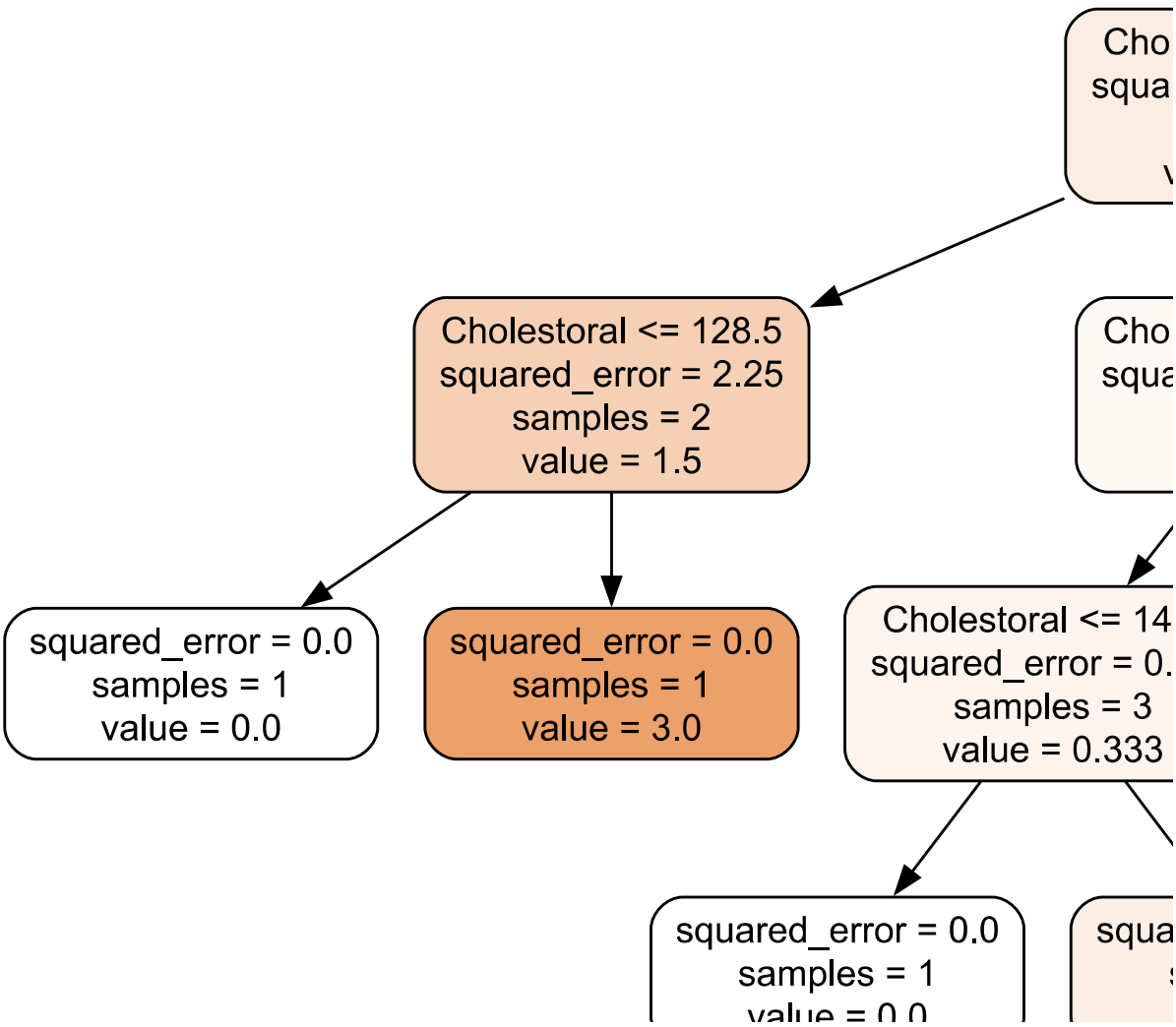
In [ ]:
```
X_grid = X_grid.reshape((len(X_grid), 1))

plt.scatter(X, y, color='red')
plt.plot(X_grid, regressor.predict(X_grid), color='blue')
plt.title('Cholestoral to Heart Disease (Decision Tree Regressor)')
plt.xlabel('Cholestoral')
plt.ylabel('Heart Disease Presence')
plt.show()
```



In [ ]:
```
from sklearn import tree
import graphviz

dot_data = tree.export_graphviz(
    regressor,
    out_file=None,
    feature_names=['Cholestoral'],
```

```
    filled=True,
    rounded=True,
)

graph = graphviz.Source(dot_data)
graph
```

Out[ ]:


Cho
squa
Cholestoral <= 128.5
squared_error = 2.25
samples = 2
value = 1.5
Cho
squa
squared_error = 0.0
samples = 1
value = 0.0
squared_error = 0.0
samples = 1
value = 3.0
Cholestoral <= 14
squared_error = 0.
samples = 3
value = 0.333
squared_error = 0.0
samples = 1
squa

DATASET

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np

# Plot a confusion matrix.
# cm is the confusion matrix, names are the names of the classes.


def plot_confusion_matrix(cm, names, title='Confusion matrix', cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(names))
    plt.xticks(tick_marks, names, rotation=45)
    plt.yticks(tick_marks, names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [ ]:
```
import pandas as pd
import seaborn as sns

data_set = pd.read_csv('./data/uci_heartDisease_changed.csv', header='infer')
data = data_set.replace('?', np.NaN)
data
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	1	145	233	1	2	150	0	2.3	3	0	6
1	67	1	4	160	286	0	2	108	1	1.5	2	3	3
2	67	1	4	120	229	0	2	129	1	2.6	2	2	7
3	37	1	3	130	250	0	0	187	0	3.5	3	0	3
4	41	0	2	130	204	0	2	172	0	1.4	1	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	45	1	1	110	264	0	0	132	0	1.2	2	0	7
299	68	1	4	144	193	1	0	141	0	3.4	2	2	7
300	57	1	4	130	131	0	0	115	1	1.2	2	1	7
301	57	0	2	130	236	0	2	174	0	0.0	2	1	3
302	38	1	3	138	175	0	0	173	0	0.0	1	1	3

303 rows × 14 columns

In [ ]:
```
data['num'] = data['num'].replace([0, 1], 'low risk')
data['num'] = data['num'].replace([2], 'risky')
data['num'] = data['num'].replace([3, 4], 'high risk')
data
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	1	145	233	1	2	150	0	2.3	3	0	6
1	67	1	4	160	286	0	2	108	1	1.5	2	3	3
2	67	1	4	120	229	0	2	129	1	2.6	2	2	7
3	37	1	3	130	250	0	0	187	0	3.5	3	0	3
4	41	0	2	130	204	0	2	172	0	1.4	1	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	45	1	1	110	264	0	0	132	0	1.2	2	0	7
299	68	1	4	144	193	1	0	141	0	3.4	2	2	7
300	57	1	4	130	131	0	0	115	1	1.2	2	1	7
301	57	0	2	130	236	0	2	174	0	0.0	2	1	3
302	38	1	3	138	175	0	0	173	0	0.0	1	1	3

303 rows × 14 columns

In [ ]:
```
# Paired plot using seaborn
sns.set()
sns.pairplot(
    data[
        [
            'age',
            'sex',
            'cp',
            'trestbps',
            'chol',
            'fbs',
            'restecg',
            'thalach',
            'exang',
            'oldpeak',
            'slope',
            'ca',
```

```
            'thal',
            'num'
        ]
    ],
    hue='num',
    diag_kind="kde"
)
```

Out[ ]: <seaborn.axisgrid.PairGrid at 0x22e062c2900>

In [ ]:
```
pd.crosstab([data['age'], data['chol']], data['num'])
```

Out[ ]:

age	chol	num	high risk	low risk	risky
29	**204**	0	1	0
34	**182**	0	1	0
	210	0	1	0
35	**183**	0	1	0
	192	0	1	0
...	**...**	...	...	...
71	**265**	0	1	0
	302	0	1	0
74	**269**	0	1	0
76	**197**	0	1	0
77	**304**	1	0	0

299 rows × 3 columns

Data Tree Classifier

In [ ]:

```
from sklearn import tree

Y = pd.DataFrame(data, columns=['num'])
X = data.drop(['num'], axis=1)

clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3)
clf = clf.fit(X, Y)

X
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	1	145	233	1	2	150	0	2.3	3	0	6
1	67	1	4	160	286	0	2	108	1	1.5	2	3	3
2	67	1	4	120	229	0	2	129	1	2.6	2	2	7
3	37	1	3	130	250	0	0	187	0	3.5	3	0	3
4	41	0	2	130	204	0	2	172	0	1.4	1	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	45	1	1	110	264	0	0	132	0	1.2	2	0	7
299	68	1	4	144	193	1	0	141	0	3.4	2	2	7
300	57	1	4	130	131	0	0	115	1	1.2	2	1	7
301	57	0	2	130	236	0	2	174	0	0.0	2	1	3
302	38	1	3	138	175	0	0	173	0	0.0	1	1	3

303 rows × 13 columns

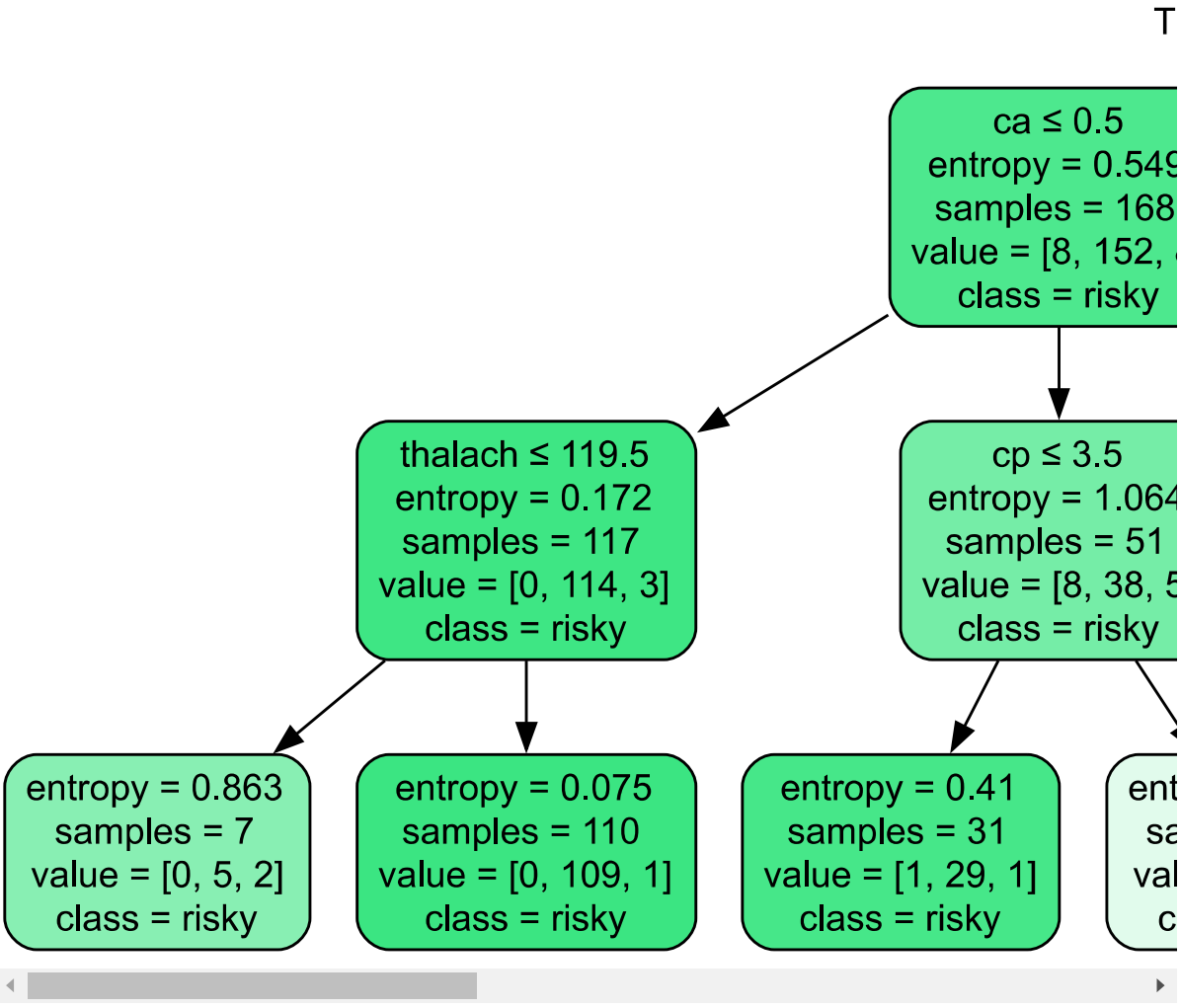
The preceding commands will extract the predictor (X) and target class (Y) attributes from the vertebrate dataset and create a decision tree classifier object using entropy as its impurity measure for splitting criterion. The decision tree class in Python sklearn library also supports using `gini` as impurity measure.

The classifier above is also constrained to generate trees with a maximum depth equals to 3. Next, the classifier is trained on the labeled data using the `fit()` function.

In [ ]:

```
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)

dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=X.columns,
                                class_names=['low risk', 'risky', 'high risk'],
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

Out[ ]:



Test Examples:

In [ ]:
```
testData = [[42, 0, 3, 150, 229, 0, 2, 171, 0, 1.9, 3, 0, 6, 'risky'],
            [65, 1, 4, 128, 126, 1, 0, 168, 1, 0.4, 2, 1, 7, 'low risk'],
            [37, 0, 2, 130, 264, 0, 1, 99, 1, 1.6, 2, 1, 3, 'low risk'],
            [55, 1, 1, 170, 220, 0, 2, 123, 0, 2.3, 2, 0, 6, 'high risk']]
testData = pd.DataFrame(testData, columns=data.columns)
testData
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	n
0	42	0	3	150	229	0	2	171	0	1.9	3	0	6	r
1	65	1	4	128	126	1	0	168	1	0.4	2	1	7	
2	37	0	2	130	264	0	1	99	1	1.6	2	1	3	
3	55	1	1	170	220	0	2	123	0	2.3	2	0	6	h

In [ ]:
```
testY = pd.DataFrame(testData, columns=['num'])
testX = testData.drop(['num'], axis=1)

predY = clf.predict(testX)
predictions = pd.concat(
    [testData['age'], pd.Series(predY, name='Predicted Class')], axis=1)
predictions
```

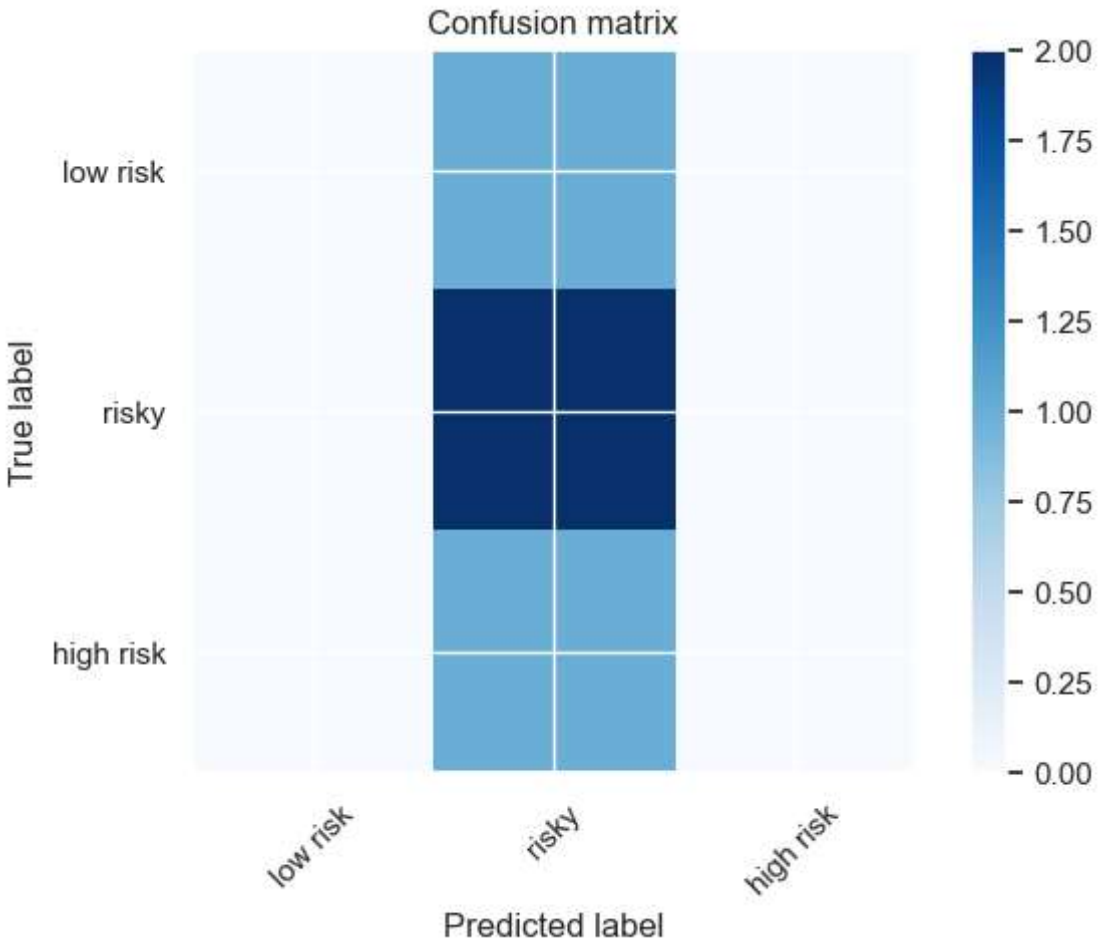
Out[ ]:

	age	Predicted Class
0	42	low risk
1	65	low risk
2	37	low risk
3	55	low risk

In [ ]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, precision_s

confusion = confusion_matrix(testY, predY)
print(confusion)

plot_confusion_matrix(confusion, data.num.unique(),
                      title='Confusion matrix', cmap=plt.cm.Blues)

print('Accuracy on test data is %.2f' % (accuracy_score(testY, predY)))
print('F1 score on test data is %.2f' %
      (f1_score(testY, predY, average='weighted')))
print('Precision Score on test data is %.2f' %
      (precision_score(testY, predY, average='weighted')))
print('Recall score on test data is %.2f' %
      (recall_score(testY, predY, average='weighted')))
print(classification_report(testY, predY))
```

```
[[0 1 0]
 [0 2 0]
 [0 1 0]]
Accuracy on test data is 0.50
F1 score on test data is 0.33
Precision Score on test data is 0.25
Recall score on test data is 0.50
              precision    recall  f1-score   support

   high risk       0.00      0.00      0.00         1
    low risk       0.50      1.00      0.67         2
       risky       0.00      0.00      0.00         1

    accuracy                           0.50         4
   macro avg       0.17      0.33      0.22         4
weighted avg       0.25      0.50      0.33         4
```

```
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\metrics\_classificatio
n.py:1471: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\metrics\_classificatio
n.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being s
et to 0.0 in labels with no predicted samples. Use `zero_division` parameter to cont
rol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\metrics\_classificatio
n.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being s
et to 0.0 in labels with no predicted samples. Use `zero_division` parameter to cont
rol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\metrics\_classificatio
n.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being s
et to 0.0 in labels with no predicted samples. Use `zero_division` parameter to cont
rol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



In [ ]:

```
from sklearn.linear_model import LogisticRegression

C = [20, 68, 90, 111, 118, 140, 177, 210, 220, 250]

LRtestAcc = []
LRtrainAcc = []

for param in C:
    clf = LogisticRegression(C=param)
    clf.fit(X, Y)
    log_reg_pred = clf.predict(testX)
    log_reg_pred_train = clf.predict(X)
    print(log_reg_pred)
    LRtestAcc.append(accuracy_score(testY, log_reg_pred))
    LRtrainAcc.append(accuracy_score(Y, log_reg_pred_train))

plt.plot(C, LRtestAcc, 'bv--', C, LRtrainAcc, 'ro--')
plt.legend(['Test Accuracy', 'Train Accuracy'])
plt.xlabel('C')
plt.xscale('log')
plt.ylabel('Accuracy')
```

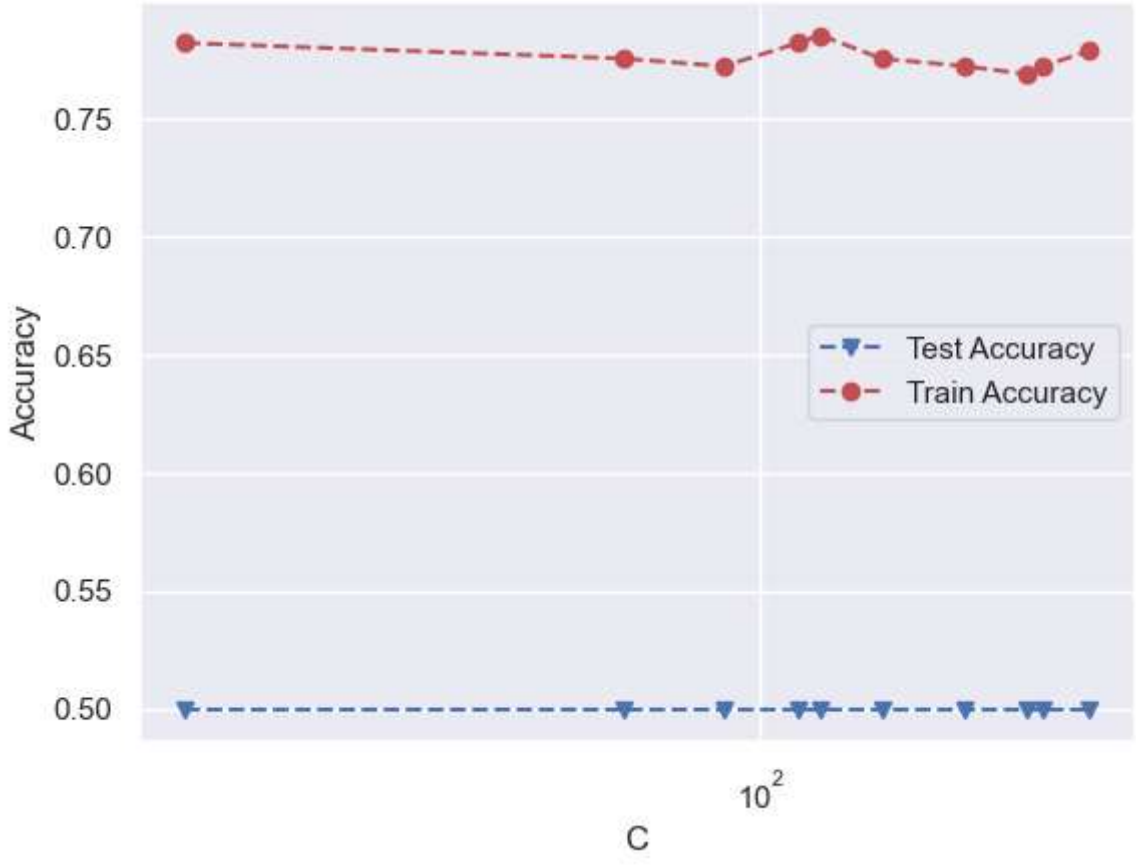
```
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\linear_model\_logistic.
py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\linear_model\_logistic.
py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\linear_model\_logistic.
py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\linear_model\_logistic.
py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
```

```
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\linear_model\_logistic.
py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\linear_model\_logistic.
py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\linear_model\_logistic.
py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\linear_model\_logistic.
py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\linear_model\_logistic.
py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
```

```
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\linear_model\_logistic.
py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[ ]: Text(0, 0.5, 'Accuracy')



Naive Bayes Classifier

In [ ]:
```
from sklearn.naive_bayes import GaussianNB

clf_NB = GaussianNB()
clf_NB.fit(X, Y)
NB_pred = clf_NB.predict(testX)
print(NB_pred)

print('Accuracy on test data is %.2f' % (accuracy_score(testY, NB_pred)))
```

```
['low risk' 'risky' 'low risk' 'low risk']
Accuracy on test data is 0.25
```

```
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
```

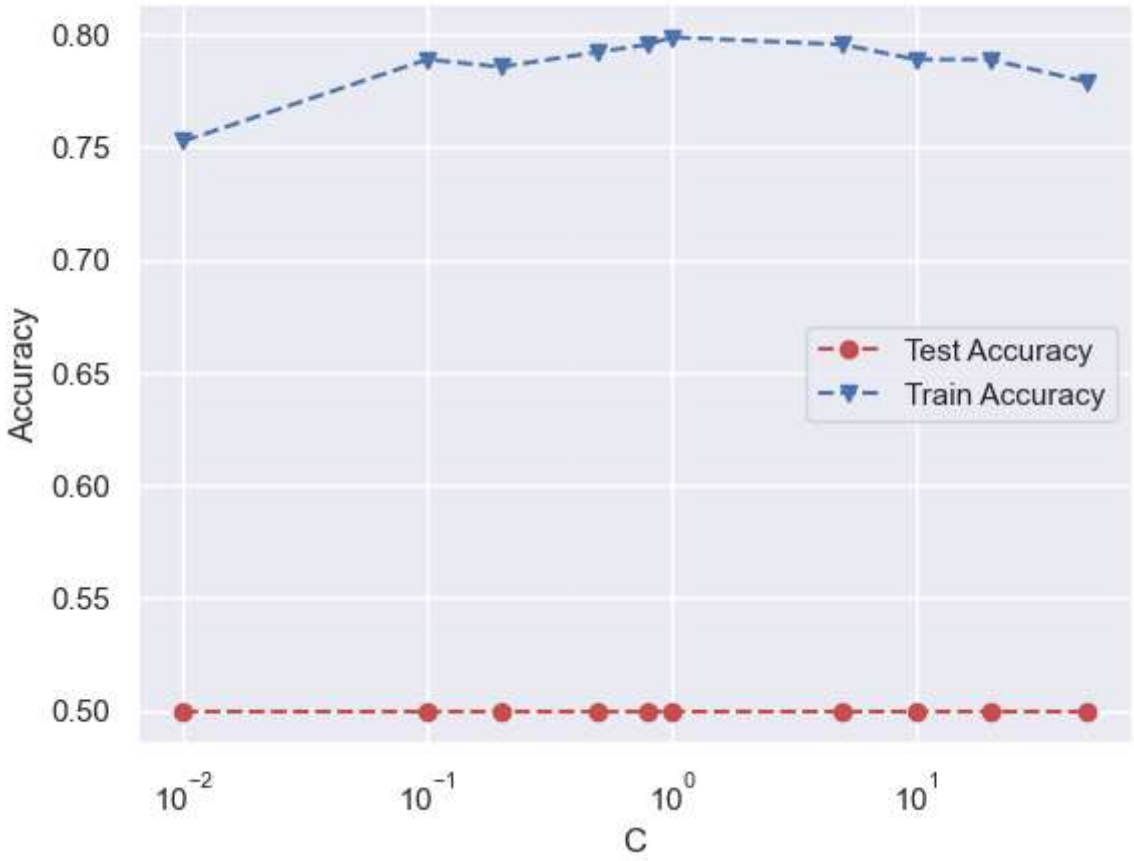
Support Vector Machine (SVM) Classifier

Linear Decision Boundary

In [ ]:
```
from sklearn.svm import SVC

C = [0.01, 0.1, 0.2, 0.5, 0.8, 1, 5, 10, 20, 50]

SVMLtestAcc = []
SVMLtrainAcc = []

for param in C:
    clf = SVC(C=param, kernel='linear')
    clf.fit(X, Y)
    svml_pred = clf.predict(testX)
    svml_pred_train = clf.predict(X)
    print(svml_pred)
    SVMLtestAcc.append(accuracy_score(testY, svml_pred))
    SVMLtrainAcc.append(accuracy_score(Y, svml_pred_train))


plt.plot(C, SVMLtestAcc, 'ro--', C, SVMLtrainAcc, 'bv--')
plt.legend(['Test Accuracy', 'Train Accuracy'])
plt.xlabel('C')
plt.xscale('log')
plt.ylabel('Accuracy')
```

```
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
['high risk' 'low risk' 'low risk' 'low risk']
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
['high risk' 'low risk' 'low risk' 'low risk']
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
['low risk' 'low risk' 'low risk' 'low risk']
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
['high risk' 'low risk' 'low risk' 'low risk']
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
['high risk' 'low risk' 'low risk' 'low risk']
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
['low risk' 'low risk' 'low risk' 'low risk']
```

```
Out[ ]: Text(0, 0.5, 'Accuracy')
```



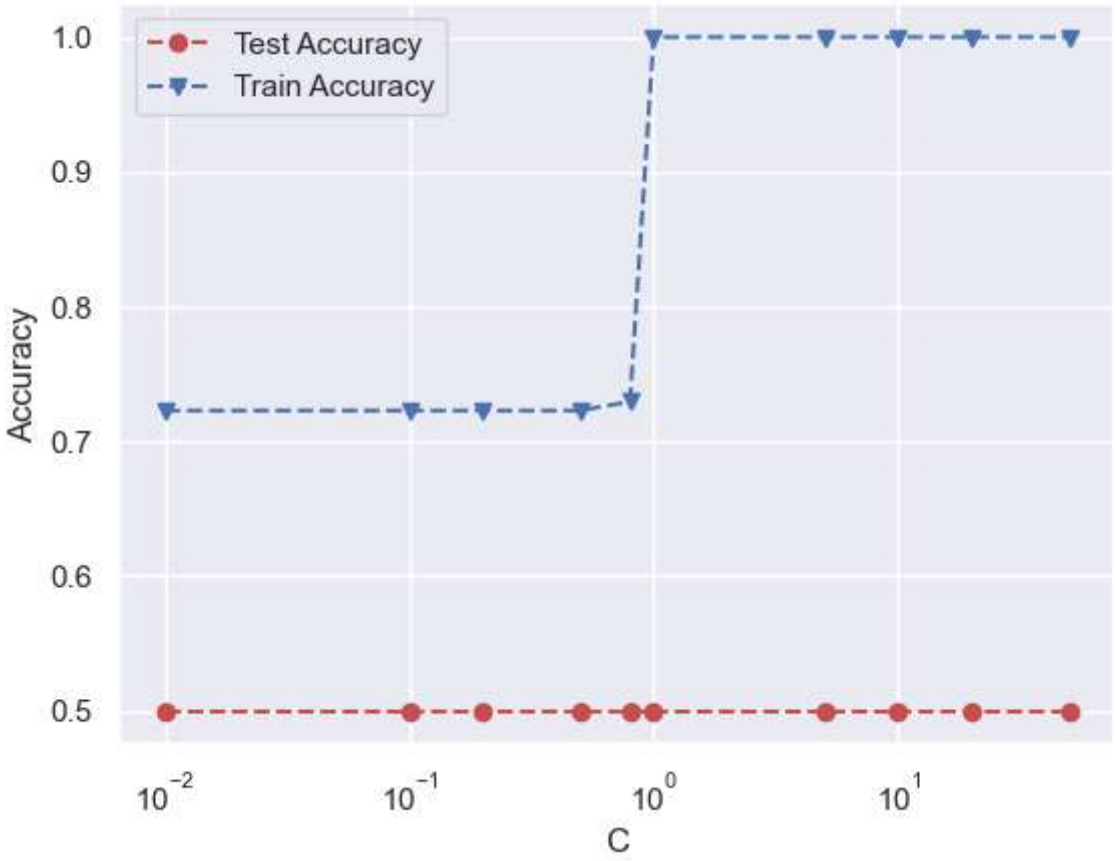
Non Linear Decision Boundary

In [ ]:

```
C = [0.01, 0.1, 0.2, 0.5, 0.8, 1, 5, 10, 20, 50]


SVMLtestAcc = []
SVMLtrainAcc = []


for param in C:
    clf = SVC(C=param, kernel='rbf', gamma='auto')
    clf.fit(X, Y)
    svml_pred = clf.predict(testX)
    svml_pred_train = clf.predict(X)
    print(svml_pred)
    SVMLtestAcc.append(accuracy_score(testY, svml_pred))
    SVMLtrainAcc.append(accuracy_score(Y, svml_pred_train))

plt.plot(C, SVMLtestAcc, 'ro--', C, SVMLtrainAcc, 'bv--')
plt.legend(['Test Accuracy', 'Train Accuracy'])
plt.xlabel('C')
plt.xscale('log')
plt.ylabel('Accuracy')
```

```
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
```

Out[ ]:
```
Text(0, 0.5, 'Accuracy')
```



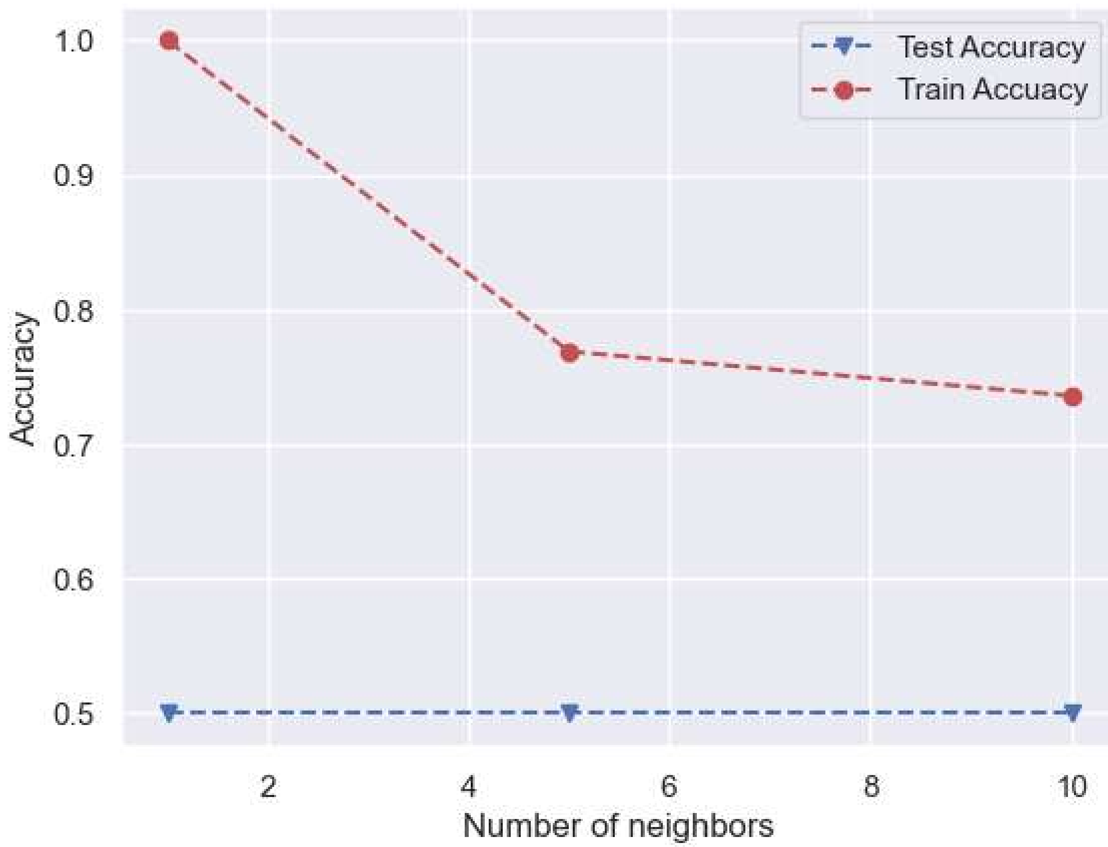
K Nearest Neighbor (KNN) Classifier

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier

numNeighbors = [1, 5, 10]
testAcc = []
trainAcc = []

for k in numNeighbors:
    clf = KNeighborsClassifier(n_neighbors=k, metric='minkowski', p=2)
    clf.fit(X, Y)
    knn_pred = clf.predict(testX)
    knn_pred_train = clf.predict(X)
    print(knn_pred)
    testAcc.append(accuracy_score(testY, knn_pred))
    trainAcc.append(accuracy_score(Y, knn_pred_train))

plt.plot(numNeighbors, testAcc, 'bv--', numNeighbors, trainAcc, 'ro--')
plt.legend(['Test Accuracy', 'Train Accuacy'])
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
```

```
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
['low risk' 'low risk' 'low risk' 'low risk']
```

```
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\neighbors\_classificati
on.py:233: DataConversionWarning: A column-vector y was passed when a 1d array was e
xpected. Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\neighbors\_classificati
on.py:233: DataConversionWarning: A column-vector y was passed when a 1d array was e
xpected. Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\neighbors\_classificati
on.py:233: DataConversionWarning: A column-vector y was passed when a 1d array was e
xpected. Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
```

Out[ ]: Text(0, 0.5, 'Accuracy')



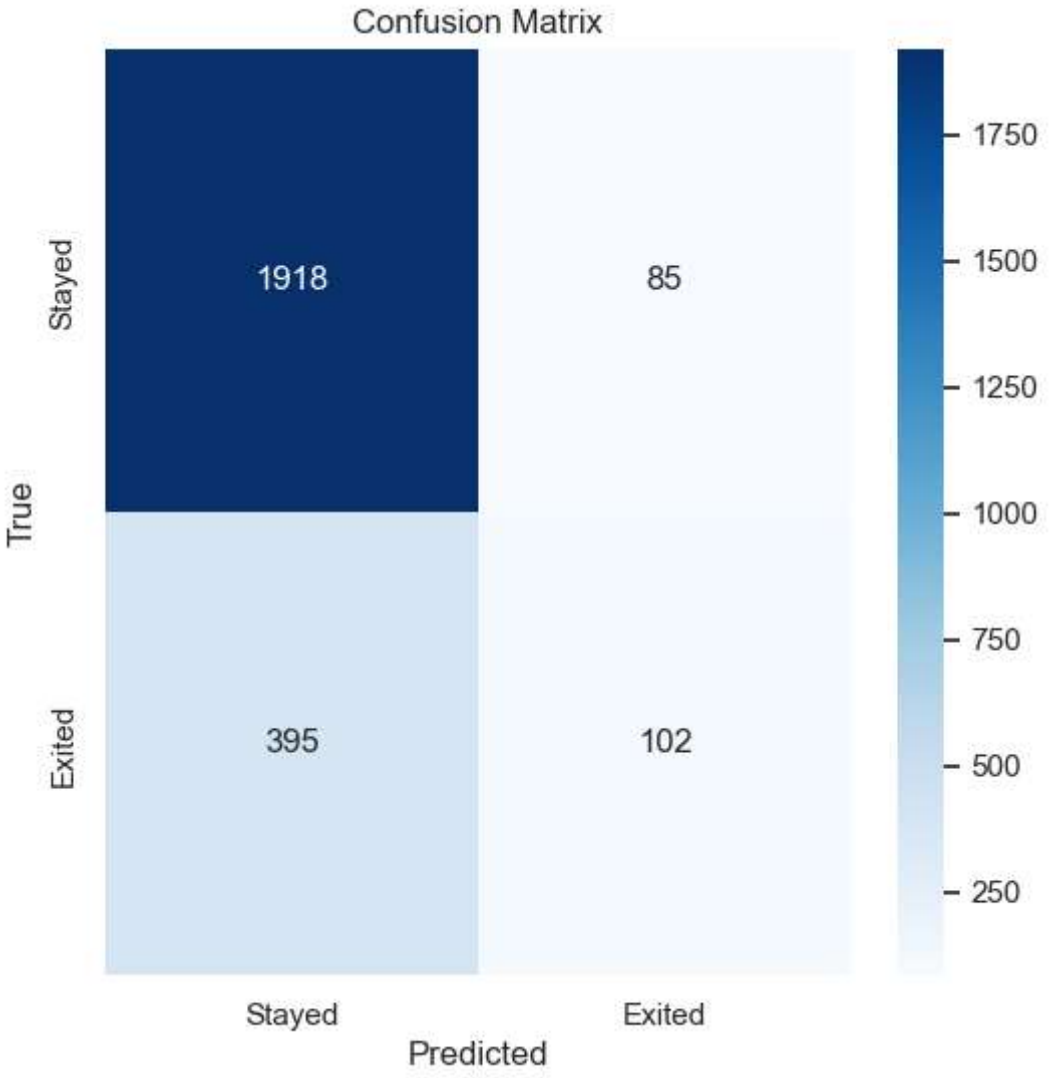
Part 2: Feature selection and Data splitting on the provided dataset

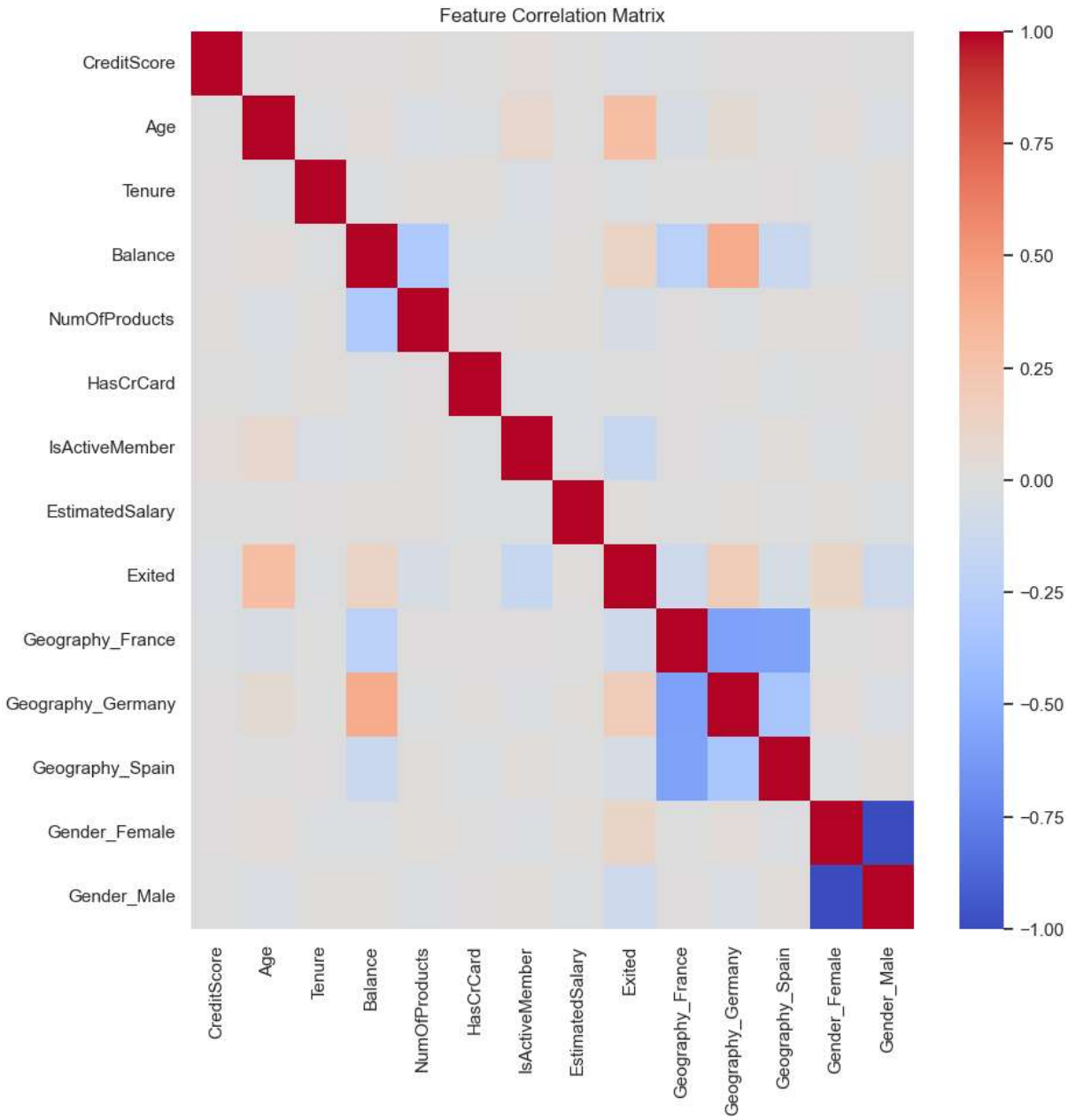
In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
```

```python
# Load the dataset
data = pd.read_csv('./data/churn_modelling.csv')

# Drop irrelevant columns if they exist
columns_to_drop = ['RowNumber', 'CustomerId', 'Surname']
for col in columns_to_drop:
    if col in data.columns:
        data = data.drop(columns=col)

# Define X (features) and y (target variable)
X = data.drop('Exited', axis=1)
y = data['Exited']

# Preprocessing: Define column transformer for scaling and encoding
numerical_cols = X.select_dtypes(include=['int64', 'float64']).columns
categorical_cols = X.select_dtypes(include=['object']).columns

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(), categorical_cols)
    ])

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42)

# Create a pipeline with preprocessing, feature selection, and logistic regression
pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('feature_selection', SelectKBest(f_classif, k=10)),  # Select top 10 features
    ('classifier', LogisticRegression())
])

# Fit the pipeline on the training data
pipeline.fit(X_train, y_train)

# Predict and evaluate on the test set
y_pred = pipeline.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model accuracy: {accuracy}")

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=[
            'Stayed', 'Exited'], yticklabels=['Stayed', 'Exited'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

# Correlation Matrix (on the original data before feature selection)
data_encoded = pd.get_dummies(data)
corr = data_encoded.corr()
```

```
plt.figure(figsize=(10, 10))
sns.heatmap(corr, annot=False, cmap='coolwarm')
plt.title('Feature Correlation Matrix')
plt.show()
```

```
Model accuracy: 0.808
```





Part 3: Running the classification models on the provided dataset

In [ ]:

```
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import os
import pandas as pd

# getting the dataset
path = "."
filename = os.path.join(path, "data/churn_modelling.csv")
df = pd.read_csv(filename, na_values=['NA', '?'])
df = df.drop(['Geography', 'Gender'], axis=1)

tdf = df[0:999]
```

In [ ]:
```
Y = pd.DataFrame(df, columns=['Exited'])
X = df.drop(['Surname', 'Exited'], axis=1)

testY = pd.DataFrame(tdf, columns=['Exited'])
testX = tdf.drop(['Surname', 'Exited'], axis=1)
```

Naive Bayes

In [ ]:
```
from sklearn.naive_bayes import GaussianNB

clf_NB = GaussianNB()
clf_NB.fit(X, Y)
NB_pred = clf_NB.predict(testX)

print('Accuracy on NB test data is %.2f' % (accuracy_score(testY, NB_pred)))
```

```
Accuracy on NB test data is 0.78
```

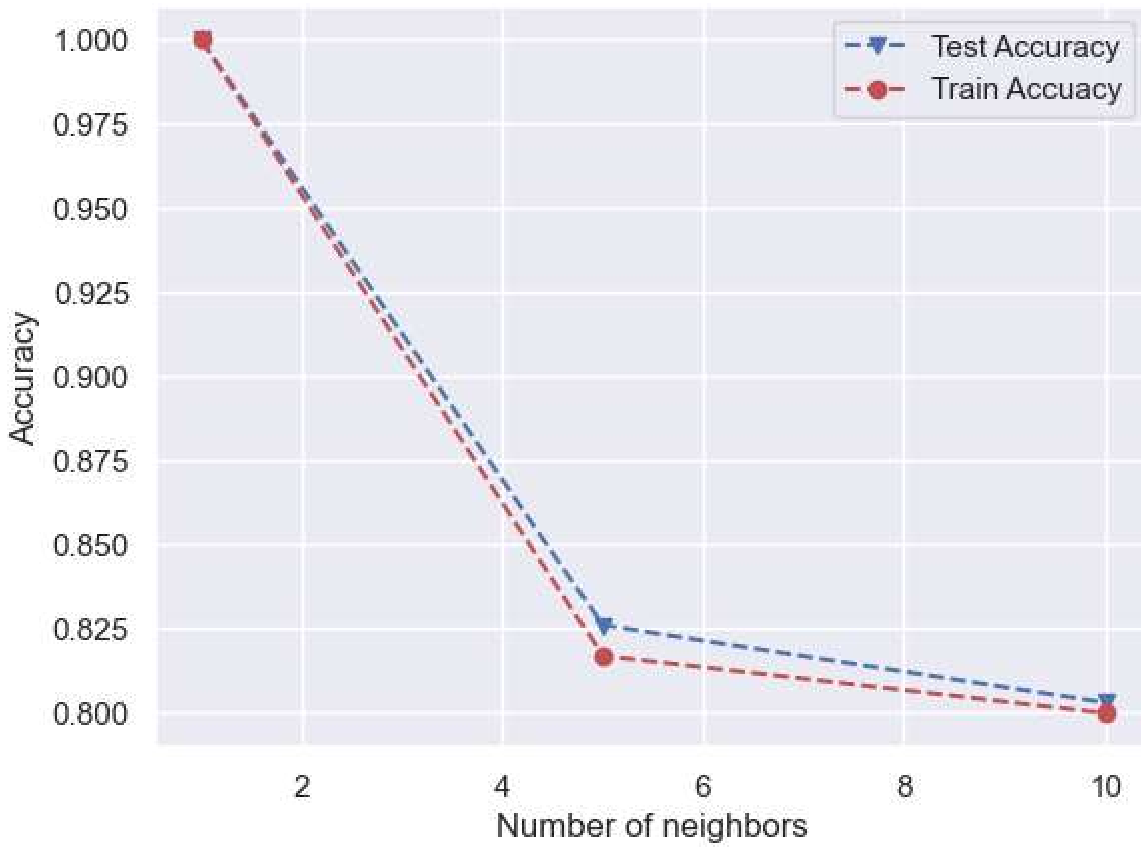
```
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

K-Nearest Neighbor

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier

numNeighbors = [1, 5, 10]
testAcc = []
trainAcc = []

for k in numNeighbors:
    clf = KNeighborsClassifier(n_neighbors=k, metric='minkowski', p=2)
    clf.fit(X, Y)
    knn_pred = clf.predict(testX)
    knn_pred_train = clf.predict(X)
    testAcc.append(accuracy_score(testY, knn_pred))
    trainAcc.append(accuracy_score(Y, knn_pred_train))

plt.plot(numNeighbors, testAcc, 'bv--', numNeighbors, trainAcc, 'ro--')
plt.legend(['Test Accuracy', 'Train Accuacy'])
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
```

```
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\neighbors\_classificati
on.py:233: DataConversionWarning: A column-vector y was passed when a 1d array was e
xpected. Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\neighbors\_classificati
on.py:233: DataConversionWarning: A column-vector y was passed when a 1d array was e
xpected. Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\neighbors\_classificati
on.py:233: DataConversionWarning: A column-vector y was passed when a 1d array was e
xpected. Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
```

Out[ ]: `Text(0, 0.5, 'Accuracy')`



In [ ]:
```python
print('Accuracy on KNN test data is %.2f' % (accuracy_score(testY, knn_pred)))
```

```
Accuracy on KNN test data is 0.80
```

Support Vector Machines

In [ ]:
```python
from sklearn.svm import SVC

C = [0.1, 0.25, 0.5, 1, 3, 5]

SVMLtestAcc = []
SVMLtrainAcc = []

# test different values of C
for param in C:
```

```python
    # we are using a linear kernel here
    clf = SVC(C=param, kernel='linear')
    # fit the model to the training data
    clf.fit(X, Y)
    # predict the test data using the model we just fit
    svml_pred = clf.predict(testX)
    # predict the training data using the model we just fit
    svml_pred_train = clf.predict(X)
    # append the accuracy score to the list
    SVMLtestAcc.append(accuracy_score(testY, svml_pred))
    # append the accuracy score to the list
    SVMLtrainAcc.append(accuracy_score(Y, svml_pred_train))
```

```
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
```

In [ ]:
```python
print('Accuracy on SVM test data is %.2f' % (accuracy_score(testY, svml_pred)))
```

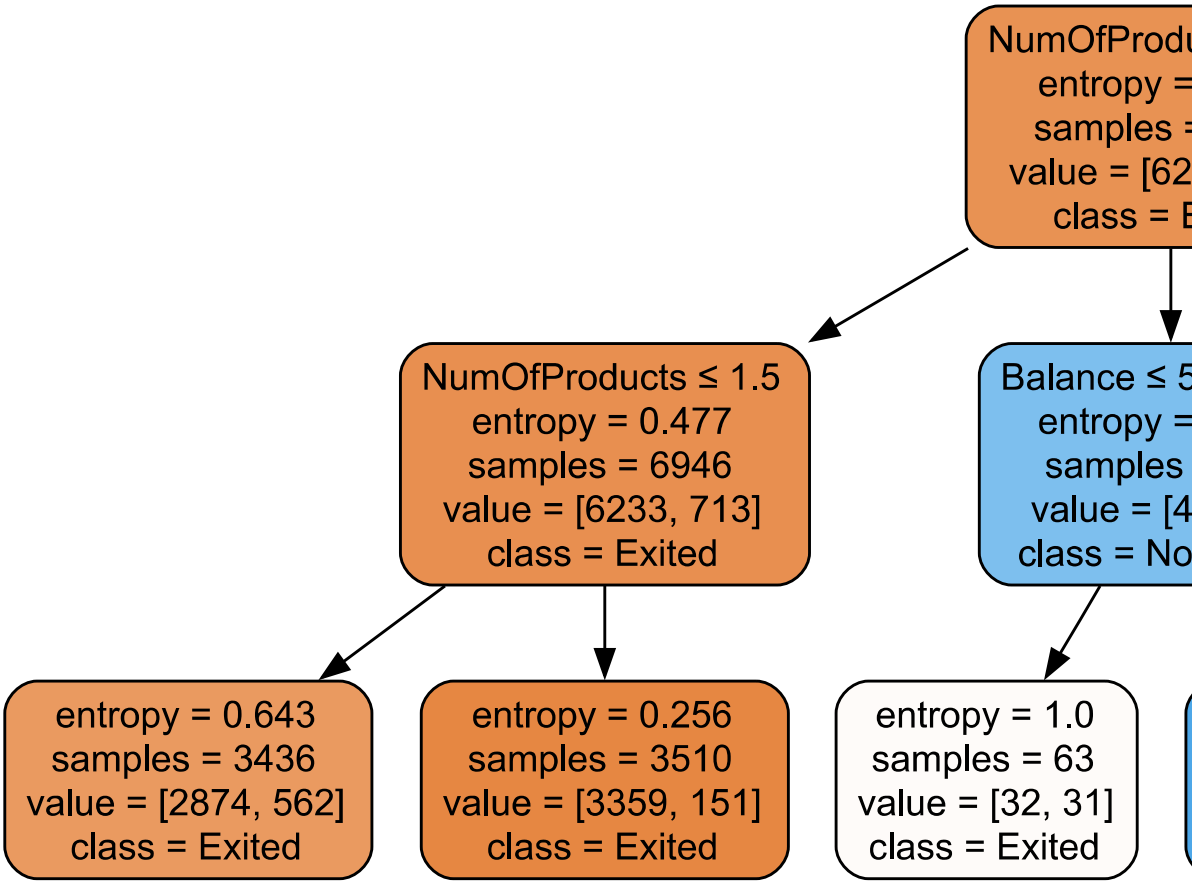
```
Accuracy on SVM test data is 0.80
```

Decision Trees

In [ ]:
```python
from sklearn import tree
import graphviz  # Adding graphviz to help graph the tree

clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3)
clf = clf.fit(X, Y)
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)

dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=X.columns,
                                class_names=['Exited', 'Not-Exited'],
                                filled=True, rounded=True,
```

```
                                    special_characters=True)
dt_graph = graphviz.Source(dot_data)
dt_graph
```

Out[ ]:



In [ ]:
```
predY = clf.predict(testX)

print('Accuracy on Decision Tree test data is %.2f' %
      (accuracy_score(testY, predY)))
```

```
Accuracy on Decision Tree test data is 0.84
```

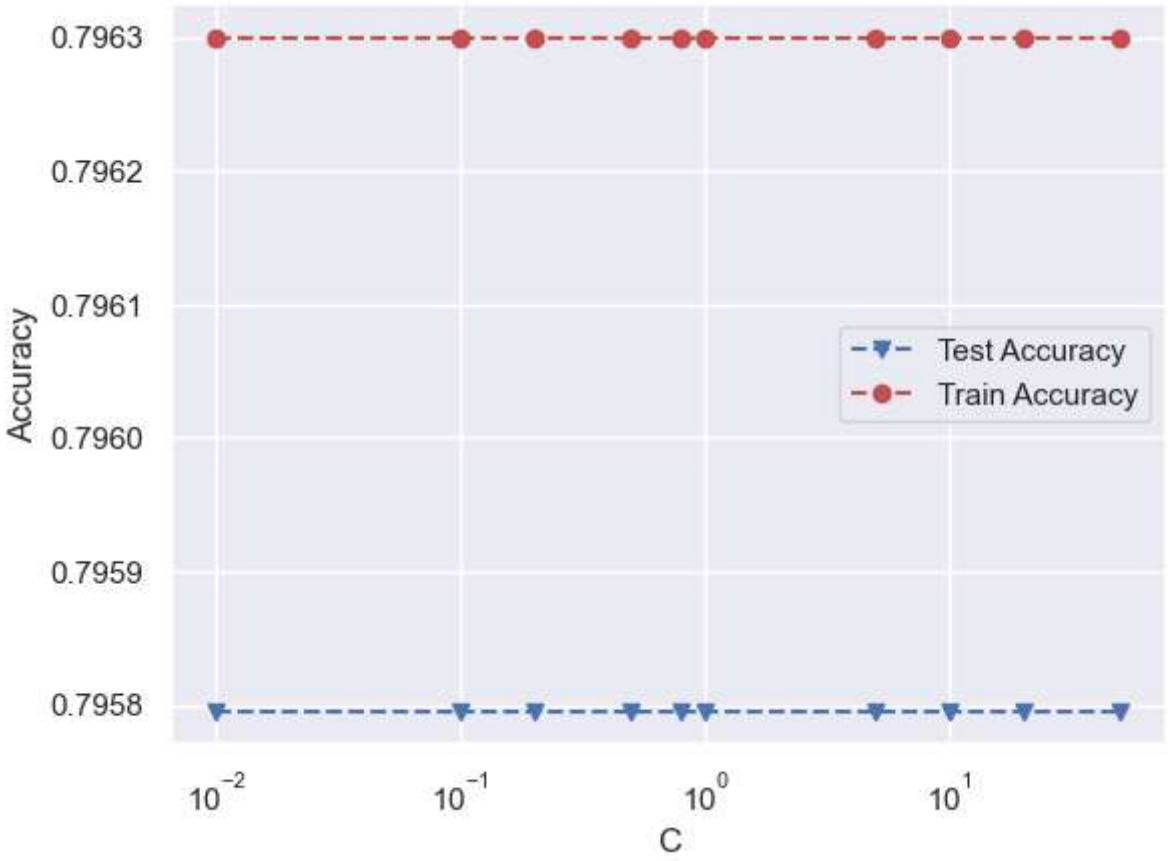
Logistic Regression

In [ ]:
```
from sklearn.linear_model import LogisticRegression

C = [0.01, 0.1, 0.2, 0.5, 0.8, 1, 5, 10, 20, 50]
```

```
LRtestAcc = []
LRtrainAcc = []

for param in C:
    clf = LogisticRegression(C=param)
    clf.fit(X, Y)
    log_reg_pred = clf.predict(testX)
    log_reg_pred_train = clf.predict(X)
    LRtestAcc.append(accuracy_score(testY, log_reg_pred))
    LRtrainAcc.append(accuracy_score(Y, log_reg_pred_train))

plt.plot(C, LRtestAcc, 'bv--', C, LRtrainAcc, 'ro--')
plt.legend(['Test Accuracy', 'Train Accuracy'])
plt.xlabel('C')
plt.xscale('log')
plt.ylabel('Accuracy')
```

```
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
d:\repo\csc177-group-project\.venv\Lib\site-packages\sklearn\utils\validation.py:118
3: 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)
```

Out[ ]: `Text(0, 0.5, 'Accuracy')`



In [ ]:
```
print('Accuracy on Logistic Regression test data is %.2f' %
      (accuracy_score(testY, log_reg_pred))
      )
```

```
Accuracy on Logistic Regression test data is 0.80
```

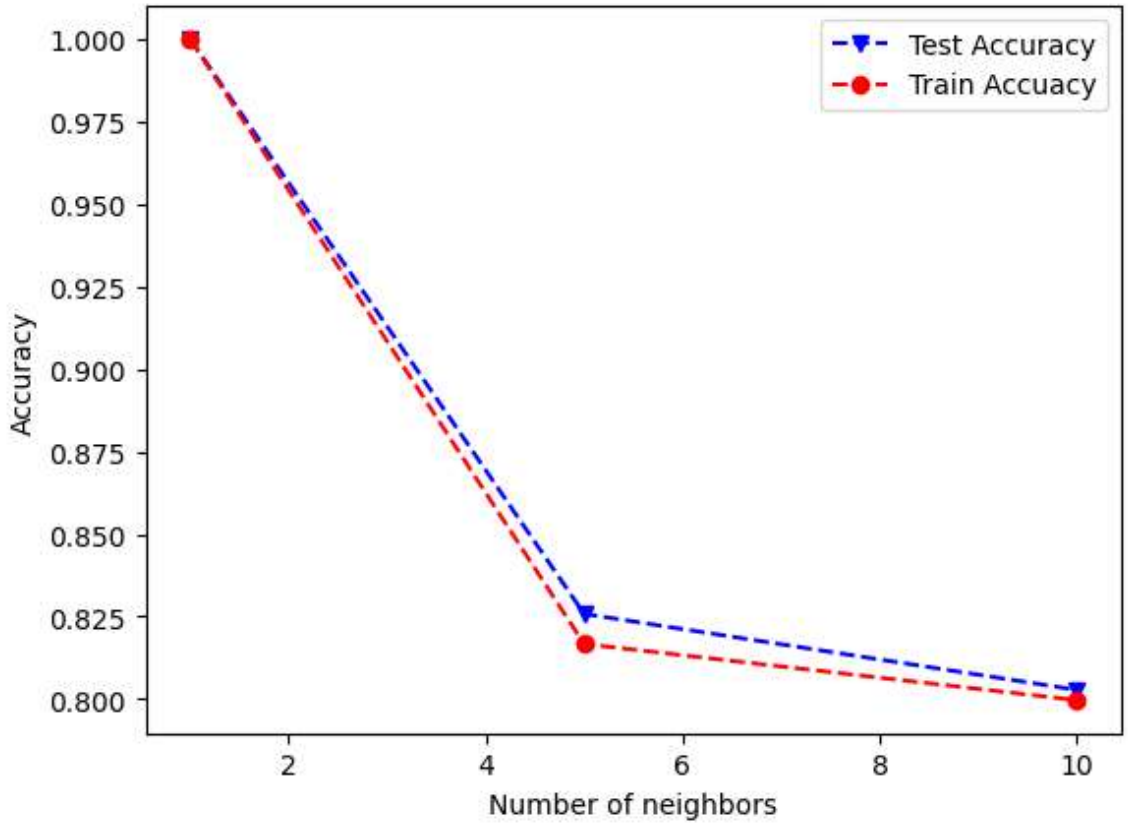
Part 4: Performing accuracy analysis and comparing the models on the provided dataset

ML Model	Accuracy on Test Set in Percent
Naive Bayes	78
KNN	80
SVM	80
DT	84
Logit	80

See Running the classification models on the provided dataset to see how the above percentages were computed.

Some graphs

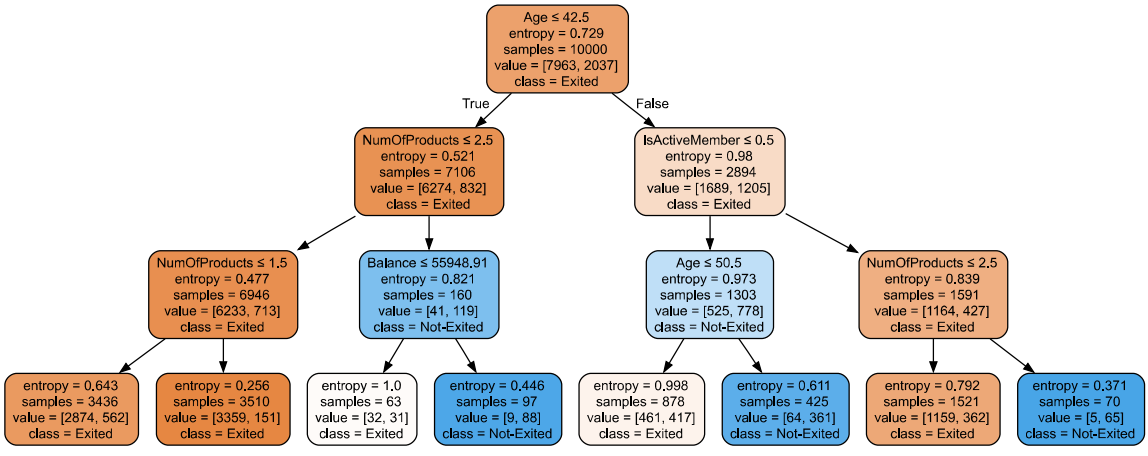
How accuracy changes based on number of neighbors:



Interpretation:

The accuracy is high with 1 neighbor because of the small sample size. Basically, these two graphs being similar shows that the test/train split was not egregious. The split did not adversely affect the training.

The decision tree



Based on our accuracy analysis, following this tree leads to the right categorization around 80% of the time, depending on the model.

The below graph (notice the y axis) shows that the test and train accuracy differences were not significant. This implies that the dataset was not split in a way that decreases the

effectiveness of training.