ML Project Code

December 9, 2018

```
In [65]: #Import required files

         import pandas as pd
         from sklearn.model_selection import train_test_split
         from scipy.cluster.hierarchy import dendrogram, ward, cut_tree
         from scipy.spatial.distance import pdist
         import matplotlib.pyplot as plt
         from sklearn.cluster import KMeans
         import scipy.spatial.distance as ssd
         from sklearn.preprocessing import OneHotEncoder
         from sklearn.linear_model import LinearRegression
         from sklearn import linear_model
         from sklearn.linear_model import Ridge
         from sklearn.linear_model import RidgeClassifier
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
         from sklearn.ensemble import RandomForestRegressor
         from sklearn.model_selection import KFold
         from sklearn.model_selection import cross_val_score
         from sklearn import metrics
         from sklearn.ensemble import GradientBoostingClassifier
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.preprocessing import PolynomialFeatures
```

```
In [66]: #load data from csv

         df = pd.read_csv('crimes.csv',sep=',')

         columns = ['year','month','day','hour','location_type','zipcode','x_coord','y_coord','o

         #columns = ['year','zipcode','offence_type','x_coord','y_coord','severity']

         # create dataframe with relevant columns

         df_2 = df[columns]
```

```
In [67]: # Enumerate output column:
```

```
crimes = pd.factorize(df_2['offence_type'])
df_2['offence_type'] = crimes[0]
crime_list = crimes[1] #create list of crime references that are coded to specific fact

# Enumerate input variables:

year = pd.factorize(df_2['year'])
df_2['year'] = year[0]
year_list = year[1]

month = pd.factorize(df_2['month'])
df_2['month'] = month[0]
month_list = month[1]

day = pd.factorize(df_2['day'])
df_2['day'] = day[0]
day_list = day[1]

hour = pd.factorize(df_2['hour'])
df_2['hour'] = hour[0]
hour_list = hour[1]

location_type = pd.factorize(df_2['location_type'])
df_2['location_type'] = location_type[0]
location_type_list = location_type[1]

# council_district = pd.factorize(df_2['council_district'])
# df_2['council_district'] = council_district[0]
# council_district_list = council_district[1]

# apd_sector = pd.factorize(df_2['apd_sector'])
# df_2['apd_sector'] = apd_sector[0]
# apd_sector_list = apd_sector[1]

# apd_district = pd.factorize(df_2['apd_district'])
# df_2['apd_district'] = apd_district[0]
# apd_district_list = apd_district[1]
```

```
/anaconda/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:4: 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/indexing.html#
  after removing the cwd from sys.path.
/anaconda/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:10: 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/indexing.html#
  # Remove the CWD from sys.path while we load stuff.
/anaconda/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:14: 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/indexing.html#

/anaconda/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:18: 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/indexing.html#
/anaconda/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:22: 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/indexing.html#
/anaconda/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:26: 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/indexing.html#
```

```
In [68]: df_2=df_2.dropna(axis=0)
         X = df_2.drop(['offence_type','severity'],axis=1).values #sets x and converts to an arr
         print(len(X))
         #print(X)

         y = df_2['severity'].values #sets y and converts to an array

         # Split the data into train and test sets for numeric encoded dataset:

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_stat
```

```
224499
```

```
In [69]: from sklearn.decomposition import PCA

         pca = PCA(4, svd_solver='randomized').fit(X_train)
         comp = pca.transform(X_train)
         comp_test = pca.transform(X_test)
         # proj = pca.inverse_transform(comp)
```

```
In [70]: #Random forest classifier

         classifier = RandomForestClassifier(n_estimators = 100, criterion = 'entropy', random_s
         classifier.fit(X_train, y_train)
         y_pred = classifier.predict(X_test)

         print(accuracy_score(y_test, y_pred))
         print(confusion_matrix(y_test, y_pred))
         print(classification_report(y_test,y_pred, target_names=crime_list))
```

```
0.7418040089086859
[[31521    50    18   704    57    18     0]
 [ 2240    32     3    75    13     0     0]
 [  942     4    10    29     7     5     0]
 [ 4961     3     1   932    22    17     1]
 [ 2218     9     5   126    27    27     0]
 [    0     0     0     0     0   785     0]
 [   37     0     0     1     0     0     0]]
                    precision    recall  f1-score   support

             Theft       0.75      0.97      0.85     32368
           Robbery       0.33      0.01      0.03      2363
        Auto Theft       0.27      0.01      0.02       997
Aggravated Assault       0.50      0.16      0.24      5937
          Burglary       0.21      0.01      0.02      2412
              Rape       0.92      1.00      0.96       785
            Murder       0.00      0.00      0.00        38

       avg / total       0.66      0.74      0.66     44900
```

```
In [7]: #Random forest classifier PCA

        classifier = RandomForestClassifier(n_estimators = 100, criterion = 'entropy', random_st
        classifier.fit(comp, y_train)
        y_pred = classifier.predict(comp_test)

        print(accuracy_score(y_test, y_pred))
        print(confusion_matrix(y_test, y_pred))
        print(classification_report(y_test,y_pred, target_names=crime_list))
```

```
0.7192992182735488
[[24207   217    75  1143   244    20     0]
 [ 1821    74    15    97    25     0     1]
 [  780    21    28    24    16     5     0]
 [ 3621    26     5   916    66    20     0]
 [ 1611    23    26   164    75    27     1]
 [    0     0     1     3     2   648     0]
 [   24     1     0     0     1     0     0]]
                    precision    recall  f1-score   support
```

```
              Theft       0.75      0.93      0.84     25906
         Auto Theft       0.20      0.04      0.06      2033
Aggravated Assault       0.19      0.03      0.05       874
           Burglary       0.39      0.20      0.26      4654
            Robbery       0.17      0.04      0.06      1927
               Rape       0.90      0.99      0.94       654
             Murder       0.00      0.00      0.00        26

        avg / total       0.63      0.72      0.66     36074
```

```
In [8]: #decision tree classifier
        kfold = KFold(n_splits=10, shuffle=True, random_state=2)
        for k in range(1,15):
                tree= DecisionTreeClassifier(max_depth=k,random_state=2)
                fold_accuracies = cross_val_score(tree, X_train, y_train, cv=kfold)
                print("k:",k)
                print("Cross-validation score:\n{}".format(fold_accuracies))
                print("average cross_validation score: {:.2f}".format(fold_accuracies.mean()))
```

```
k: 1
Cross-validation score:
[0.73742204 0.73284823 0.72619543 0.73534304 0.73562024 0.72959113
 0.73241389 0.73574052 0.73587913 0.73185945]
average cross_validation score: 0.73
k: 2
Cross-validation score:
[0.73742204 0.73284823 0.72619543 0.73527374 0.73562024 0.72959113
 0.73241389 0.73574052 0.73587913 0.73185945]
average cross_validation score: 0.73
k: 3
Cross-validation score:
[0.73749134 0.73284823 0.72619543 0.73527374 0.73562024 0.72966043
 0.73241389 0.73574052 0.73601774 0.73185945]
average cross_validation score: 0.73
k: 4
Cross-validation score:
[0.73749134 0.73284823 0.72619543 0.73527374 0.73562024 0.72959113
 0.73241389 0.73574052 0.73601774 0.73179014]
average cross_validation score: 0.73
k: 5
Cross-validation score:
[0.73832294 0.73284823 0.72619543 0.73527374 0.73562024 0.73014553
 0.73303763 0.73574052 0.73587913 0.7326218 ]
average cross_validation score: 0.73
k: 6
```

```
Cross-validation score:
[0.73804574 0.73367983 0.72674983 0.73541234 0.73652114 0.73090783
 0.73276041 0.7368494  0.73643357 0.73276041]
average cross_validation score: 0.73
k: 7
Cross-validation score:
[0.73797644 0.73381843 0.72681913 0.73534304 0.73652114 0.73139293
 0.73282972 0.73657218 0.73657218 0.73289902]
average cross_validation score: 0.73
k: 8
Cross-validation score:
[0.73887734 0.73451143 0.72875953 0.73707554 0.73742204 0.73173943
 0.73622566 0.73788897 0.73795828 0.73671079]
average cross_validation score: 0.74
k: 9
Cross-validation score:
[0.73970894 0.73575884 0.72896743 0.73742204 0.73728344 0.73409563
 0.73574052 0.73816619 0.73892855 0.73740384]
average cross_validation score: 0.74
k: 10
Cross-validation score:
[0.73846154 0.73562024 0.72875953 0.73721414 0.73555094 0.73298683
 0.7339386  0.7382355  0.73872063 0.73601774]
average cross_validation score: 0.74
k: 11
Cross-validation score:
[0.73409563 0.73520444 0.72598753 0.73513514 0.73374913 0.73215523
 0.73338416 0.73698801 0.73608705 0.73483956]
average cross_validation score: 0.73
k: 12
Cross-validation score:
[0.73153153 0.73014553 0.72356202 0.73146223 0.73146223 0.72772003
 0.72957239 0.73456234 0.73428512 0.73359207]
average cross_validation score: 0.73
k: 13
Cross-validation score:
[0.72806653 0.72910603 0.71898822 0.72626473 0.72543313 0.72460152
 0.72458244 0.72957239 0.72853282 0.72624576]
average cross_validation score: 0.73
k: 14
Cross-validation score:
[0.72127512 0.72404712 0.71330561 0.72072072 0.71850312 0.71961192
 0.71869152 0.7225726  0.7225726  0.71876083]
average cross_validation score: 0.72
```

```
In [9]: #Decision tree on test data
```

```
        tree_train = DecisionTreeClassifier(max_depth=6,random_state=2).fit(X_train, y_train)
        y_predicted = tree_train.predict(X_test)
        curTestAccuracy = tree_train.score(X_test, y_test)
        print(curTestAccuracy)
        print(metrics.classification_report(y_predicted, y_test))
```

```
0.7362366247158618
             precision    recall  f1-score   support

          1       1.00      0.73      0.85     35277
          2       0.00      0.00      0.00         0
          3       0.00      0.00      0.00         1
          4       0.01      0.57      0.02        68
          5       0.00      0.00      0.00         0
          6       1.00      0.90      0.95       728
          7       0.00      0.00      0.00         0

avg / total       1.00      0.74      0.85     36074
```

```
/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/metrics/classification.py:1137: Undefine
  'recall', 'true', average, warn_for)
```

```
In [71]: # train ridge regression

         ridge_model = linear_model.Ridge(alpha = 0.3)
         ridge_model.fit(X_train, y_train)
         ridge_predicted = ridge_model.predict(X_test)
```

```
In [72]: #ridge classifier
         ridge = RidgeClassifier().fit(X_train, y_train)
         y_ridge = ridge.predict(X_test)
         curTestAccuracy = ridge.score(X_test, y_test)
         print(curTestAccuracy)
         print(metrics.classification_report(y_ridge, y_test))
```

```
0.7379732739420936
             precision    recall  f1-score   support

          1       1.00      0.73      0.85     44048
          2       0.00      0.00      0.00         0
          3       0.00      0.00      0.00         0
          4       0.00      0.00      0.00         0
          5       0.00      0.00      0.00         0
          6       1.00      0.92      0.96       852
          7       0.00      0.00      0.00         0
```

```
avg / total       1.00      0.74      0.85     44900
```

```
/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/metrics/classification.py:1137: Undefine
  'recall', 'true', average, warn_for)
```

```
In [48]: #normalize the data
         from sklearn.preprocessing import MinMaxScaler

         mms=MinMaxScaler()
         X_train_norm = mms.fit_transform(X_train)
         X_test_norm = mms.fit_transform(X_test)
```

```
In [49]: #Question 2b Euclidian
         from sklearn.neighbors import KNeighborsClassifier
         import matplotlib.pyplot as plt
         %matplotlib inline

         def plotNumNeighborsAccuracy():
             training_accuracy=[]
             test_accuracy=[]
             neighbor_settings = range(1,7)
             for curKvalue in neighbor_settings:
                 #Build the model
                 clf = KNeighborsClassifier(n_neighbors=curKvalue)
                 fold_accuracies = cross_val_score(clf, X_train_norm, y_train, cv=kfold)
                 print("Cross-validation score:\n{}".format(fold_accuracies))
                 print("average cross_validation score: {:.2f}".format(fold_accuracies.mean()))
```

```
In [50]: plotNumNeighborsAccuracy()
```

```
Cross-validation score:
[0.56633081 0.59302814 0.59596808 0.58378832 0.57328853 0.58672827
 0.56404872 0.57034859 0.5800084  0.5749685 ]
average cross_validation score: 0.58
Cross-validation score:
[0.66708648 0.69046619 0.67744645 0.68584628 0.66190676 0.6900462
 0.65896682 0.68038639 0.67660647 0.67198656]
average cross_validation score: 0.68
Cross-validation score:
[0.67464316 0.68836623 0.67912642 0.68962621 0.66400672 0.69088618
 0.66274675 0.68962621 0.67576648 0.67786644]
average cross_validation score: 0.68
Cross-validation score:
[0.68178002 0.70558589 0.68542629 0.69802604 0.67366653 0.70222596
 0.67198656 0.70096598 0.69088618 0.69088618]
average cross_validation score: 0.69
```

```
Cross-validation score:
[0.70025189 0.71146577 0.69424612 0.70180596 0.68416632 0.71608568
 0.67576648 0.70306594 0.6900462  0.69256615]
average cross_validation score: 0.70
Cross-validation score:
[0.7031906  0.71650567 0.70138597 0.70852583 0.68710626 0.72280554
 0.68164637 0.70558589 0.70096598 0.69970601]
average cross_validation score: 0.70
```

```
In [73]: # train polynomial feature model

         poly_features = PolynomialFeatures(degree=2, include_bias=False)
         X_poly = poly_features.fit_transform(X)
         print(X.shape)
         print(X_poly.shape)


         X_polyTrain, X_polyTest, y_polyTrain, y_polyTest = train_test_split(X_poly, y, random_s
         poly_model = linear_model.LinearRegression().fit(X_polyTrain, y_polyTrain)
         y_polyPredicted = poly_model.predict(X_polyTest)
         curTestAccuracy = poly_model.score(X_polyTest, y_polyTest)
         print(curTestAccuracy)
```

```
(224499, 8)
(224499, 44)
0.15513881704609067
```

```
In [74]: # linear regression model
         from sklearn import linear_model
         lr_model = linear_model.LinearRegression().fit(X_train,y_train)
         # predicted Vs actual values
         y_predicted = lr_model.intercept_ + lr_model.coef_*X_test
```

```
In [108]: from sklearn.preprocessing import StandardScaler

          stdsc = StandardScaler()
          stdsc.fit(X_train)
          x_train_std = stdsc.transform(X_train)
          x_test_std = stdsc.transform(X_test)
```

```
In [105]: from sklearn.neural_network import MLPClassifier

          mlp = MLPClassifier(random_state=42)
          mlp.fit(x_train_std, y_train)
          print("Accuracy on training set: {:.2f}".format(mlp.score(x_train_std, y_train)))
          print("Accuracy on test set: {:.2f}".format(mlp.score(x_test_std, y_test)))
```

```
Accuracy on training set: 0.73
Accuracy on test set: 0.74
```

```
In [106]: for i in range(1,11):
              mlp2 = MLPClassifier(activation='tanh', hidden_layer_sizes=[i])
              mlp2.fit(x_train_std, y_train)
              print("Accuracy on training set: {:.2f}".format(mlp2.score(x_train_std, y_train)))
              print("Accuracy on test set: {:.2f}".format(mlp2.score(x_test_std, y_test)))
```

```
Accuracy on training set: 0.73
Accuracy on test set: 0.74
Accuracy on training set: 0.73
Accuracy on test set: 0.74
Accuracy on training set: 0.73
Accuracy on test set: 0.73
Accuracy on training set: 0.73
Accuracy on test set: 0.73
Accuracy on training set: 0.73
Accuracy on test set: 0.74
Accuracy on training set: 0.73
Accuracy on test set: 0.74
Accuracy on training set: 0.73
Accuracy on test set: 0.74
Accuracy on training set: 0.73
Accuracy on test set: 0.74
```

```
/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/neural_network/multilayer_perceptron.py:
  warnings.warn("Training interrupted by user.")
```

```
Accuracy on training set: 0.73
Accuracy on test set: 0.73
Accuracy on training set: 0.73
Accuracy on test set: 0.73
```

```
In [61]: import numpy as np
         import pandas as pd
         from sklearn.datasets import load_digits
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LogisticRegression
         from sklearn.cluster import KMeans
         from sklearn.metrics import accuracy_score
         from sklearn.svm import SVC
         from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

         class cluster():
             def load(self,data):

                 crimes = pd.factorize(data['offence_type'])
                 df_2['offence_type'] = crimes[0]
                 crime_list = crimes[1]

                 location_type = pd.factorize(data['location_type'])
                 data['location_type'] = location_type[0]
                 location_type_list = location_type[1]

                 X = data
                 y = data['offence_type'].values

                 self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(X, y, t



             def __init__(self, data):
                 self.load(data)


             def classify(self, model=LogisticRegression(random_state=42)):
                 model.fit(self.X_train, self.y_train)
                 y_pred = model.predict(self.X_test)

                 print(classification_report(self.y_test,y_pred, target_names=crime_list))
                 print('Accuracy: {}'.format(accuracy_score(self.y_test, y_pred)))



             def Kmeans(self, output='include'):
                 n_clusters = len(np.unique(self.y_train))
                 clf = KMeans(n_clusters = n_clusters, random_state=42)
                 clf.fit(self.X_train)
                 y_labels_train = clf.labels_
                 y_labels_test = clf.predict(self.X_test)
                 if output == 'include':
                     self.X_train['km_clust'] = y_labels_train
                     self.X_test['km_clust'] = y_labels_test
                 elif output == 'exclude':
                     self.X_train = y_labels_train[:, np.newaxis]
                     self.X_test = y_labels_test[:, np.newaxis]
                 else:
                     raise ValueError('output should be either add or replace')
                 return self

In [38]: df = pd.read_csv('crimes.csv',sep=',')
```

```
columns = ['year','month','day','hour','location_type','zipcode','x_coord','y_coord','o

df_2 = df[columns]
df_2 = df_2.dropna(axis=0)
cluster(df_2).Kmeans(output='include').classify()
```

```
/anaconda/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:46: 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/indexing.html#
/anaconda/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:47: 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/indexing.html#
```

```
                    precision    recall  f1-score   support

             Theft       0.73      1.00      0.85     48388
        Auto Theft       0.00      0.00      0.00      1556
Aggravated Assault       0.00      0.00      0.00      3583
          Burglary       0.00      0.00      0.00      3480
           Robbery       0.00      0.00      0.00      9014
              Rape       0.89      1.00      0.94      1149
            Murder       0.00      0.00      0.00        47

       avg / total       0.54      0.74      0.63     67217

Accuracy: 0.7364952318609875
```

```
/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/metrics/classification.py:1135: Undefine
  'precision', 'predicted', average, warn_for)
```

```
In [40]: cluster(df_2).Kmeans(output='include').classify(model=RidgeClassifier())
```

```
                    precision    recall  f1-score   support

             Theft       0.90      1.00      0.95     48388
        Auto Theft       0.00      0.00      0.00      1556
Aggravated Assault       0.00      0.00      0.00      3583
          Burglary       0.00      0.00      0.00      3480
           Robbery       0.72      1.00      0.84      9014
              Rape       0.93      1.00      0.96      1149
            Murder       0.00      0.00      0.00        47
```

```
       avg / total       0.76      0.87      0.81     67217

Accuracy: 0.870434562685035


/anaconda/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:46: 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/indexing.html#
/anaconda/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:47: 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/indexing.html#
/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/metrics/classification.py:1135: Undefine
  'precision', 'predicted', average, warn_for)
```