**Predict Crime Category - KNN**

In [88]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
% matplotlib inline
```

In [89]:
```
df = pd.read_csv("train_crime.csv")
```

In [90]:
```
df.head()
```

Out[90]:

| | Dates | Category | Descript | DayOfWeek | PdDistrict | Resolution | Address | X | Y |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2015-05-13 23:53:00 | WARRANTS | WARRANT ARREST | Wednesday | NORTHERN | ARREST, BOOKED | OAK ST / LAGUNA ST | -122.425892 | 37.774599 |
| **1** | 2015-05-13 23:53:00 | OTHER OFFENSES | TRAFFIC VIOLATION ARREST | Wednesday | NORTHERN | ARREST, BOOKED | OAK ST / LAGUNA ST | -122.425892 | 37.774599 |
| **2** | 2015-05-13 23:33:00 | OTHER OFFENSES | TRAFFIC VIOLATION ARREST | Wednesday | NORTHERN | ARREST, BOOKED | VANNESS AV / GREENWICH ST | -122.424363 | 37.800414 |
| **3** | 2015-05-13 23:30:00 | LARCENY/THEFT | GRAND THEFT FROM LOCKED AUTO | Wednesday | NORTHERN | NONE | 1500 Block of LOMBARD ST | -122.426995 | 37.800873 |
| **4** | 2015-05-13 23:30:00 | LARCENY/THEFT | GRAND THEFT FROM LOCKED AUTO | Wednesday | PARK | NONE | 100 Block of BRODERICK ST | -122.438738 | 37.771541 |

In [91]:
```
df.shape
```

Out[91]:
```
(878049, 9)
```

Understanding the data It's important to understand all the columns before we move further. Train data has the following columns: Dates - timestamp of the crime incident Category - category of the crime incident (only in train.csv). This is the target variable you are going to predict. Descript - detailed description of the crime incident (only in train.csv) DayOfWeek - the day of the week PdDistrict - name of the Police Department District Resolution - how the crime incident was resolved (only in train.csv) Address - the approximate street address of the crime incident X - Longitude Y - Latitude

In [92]:
```
df.columns.isnull()
```

Out[92]:
```
array([False, False, False, False, False, False, False, False, False])
```

In [93]:
```
##### Target value

target = df['Category'].unique()
print(target)

# There are multiple Categorical values
```

```
['WARRANTS' 'OTHER OFFENSES' 'LARCENY/THEFT' 'VEHICLE THEFT' 'VANDALISM'
 'NON-CRIMINAL' 'ROBBERY' 'ASSAULT' 'WEAPON LAWS' 'BURGLARY'
 'SUSPICIOUS OCC' 'DRUNKENNESS' 'FORGERY/COUNTERFEITING' 'DRUG/NARCOTIC'
 'STOLEN PROPERTY' 'SECONDARY CODES' 'TRESPASS' 'MISSING PERSON' 'FRAUD'
 'KIDNAPPING' 'RUNAWAY' 'DRIVING UNDER THE INFLUENCE'
 'SEX OFFENSES FORCIBLE' 'PROSTITUTION' 'DISORDERLY CONDUCT' 'ARSON'
 'FAMILY OFFENSES' 'LIQUOR LAWS' 'BRIBERY' 'EMBEZZLEMENT' 'SUICIDE'
 'LOITERING' 'SEX OFFENSES NON FORCIBLE' 'EXTORTION' 'GAMBLING'
 'BAD CHECKS' 'TREA' 'RECOVERED VEHICLE' 'PORNOGRAPHY/OBSCENE MAT']
```

# Lets read test data

```
In [94]: df_test = pd.read_csv("test_crime.csv")
         df_test.head()
```

Out[94]:

|  | Id | Dates | DayOfWeek | PdDistrict | Address | X | Y |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 2015-05-10 23:59:00 | Sunday | BAYVIEW | 2000 Block of THOMAS AV | -122.399588 | 37.735051 |
| **1** | 1 | 2015-05-10 23:51:00 | Sunday | BAYVIEW | 3RD ST / REVERE AV | -122.391523 | 37.732432 |
| **2** | 2 | 2015-05-10 23:50:00 | Sunday | NORTHERN | 2000 Block of GOUGH ST | -122.426002 | 37.792212 |
| **3** | 3 | 2015-05-10 23:45:00 | Sunday | INGLESIDE | 4700 Block of MISSION ST | -122.437394 | 37.721412 |
| **4** | 4 | 2015-05-10 23:45:00 | Sunday | INGLESIDE | 4700 Block of MISSION ST | -122.437394 | 37.721412 |

```
In [95]: df_test.shape

Out[95]: (884262, 7)
```

```
In [96]: df_test.columns.isnull()

Out[96]: array([False, False, False, False, False, False, False])
```

## Changing Category, Days, District to numerical values

```
In [97]: # Category data
         data_dict = {}
         count = 1
         for data in target:
             data_dict[data] = count
             count += 1
         df["Category"] = df["Category"].replace(data_dict)
```

```
In [98]: df["DayOfWeek"].unique()

Out[98]: array(['Wednesday', 'Tuesday', 'Monday', 'Sunday', 'Saturday', 'Friday',
                'Thursday'], dtype=object)
```

```
In [99]: # Day data
         data_week_dict = {
             "Monday": 1,
             "Tuesday":2,
             "Wednesday":3,
             "Thursday":4,
             "Friday":5,
             "Saturday":6,
             "Sunday":7
         }
         df["DayOfWeek"] = df["DayOfWeek"].replace(data_week_dict)
```

```
In [100]: df.columns

Out[100]: Index(['Dates', 'Category', 'Descript', 'DayOfWeek', 'PdDistrict',
                 'Resolution', 'Address', 'X', 'Y'],
                dtype='object')
```

In [101]:

```
# District Data
district = df["PdDistrict"].unique()
data_district = {}
count = 1
for data in district:
    data_district[data] = count
    count += 1
df["PdDistrict"] = df["PdDistrict"].replace(data_district)
```

In [102]:

```
df.head()
```

Out[102]:

| | Dates | Category | Descript | DayOfWeek | PdDistrict | Resolution | Address | X | Y |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2015-05-13 23:53:00 | 1 | WARRANT ARREST | 3 | 1 | ARREST, BOOKED | OAK ST / LAGUNA ST | -122.425892 | 37.774599 |
| **1** | 2015-05-13 23:53:00 | 2 | TRAFFIC VIOLATION ARREST | 3 | 1 | ARREST, BOOKED | OAK ST / LAGUNA ST | -122.425892 | 37.774599 |
| **2** | 2015-05-13 23:33:00 | 2 | TRAFFIC VIOLATION ARREST | 3 | 1 | ARREST, BOOKED | VANNESS AV / GREENWICH ST | -122.424363 | 37.800414 |
| **3** | 2015-05-13 23:30:00 | 3 | GRAND THEFT FROM LOCKED AUTO | 3 | 1 | NONE | 1500 Block of LOMBARD ST | -122.426995 | 37.800873 |
| **4** | 2015-05-13 23:30:00 | 3 | GRAND THEFT FROM LOCKED AUTO | 3 | 2 | NONE | 100 Block of BRODERICK ST | -122.438738 | 37.771541 |

In [103]:

```
data_week_dict = {
    "Monday": 1,
    "Tuesday":2,
    "Wednesday":3,
    "Thursday":4,
    "Friday":5,
    "Saturday":6,
    "Sunday":7
}
df_test["DayOfWeek"] = df_test["DayOfWeek"].replace(data_week_dict)
```

In [104]:

```
district = df_test["PdDistrict"].unique()
data_district = {}
count = 1
for data in district:
    data_district[data] = count
    count += 1
df_test["PdDistrict"] = df_test["PdDistrict"].replace(data_district)
```

In [105]:

```
df_test.head()
```

Out[105]:

| | Id | Dates | DayOfWeek | PdDistrict | Address | X | Y |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 2015-05-10 23:59:00 | 7 | 1 | 2000 Block of THOMAS AV | -122.399588 | 37.735051 |
| **1** | 1 | 2015-05-10 23:51:00 | 7 | 1 | 3RD ST / REVERE AV | -122.391523 | 37.732432 |
| **2** | 2 | 2015-05-10 23:50:00 | 7 | 2 | 2000 Block of GOUGH ST | -122.426002 | 37.792212 |
| **3** | 3 | 2015-05-10 23:45:00 | 7 | 3 | 4700 Block of MISSION ST | -122.437394 | 37.721412 |
| **4** | 4 | 2015-05-10 23:45:00 | 7 | 3 | 4700 Block of MISSION ST | -122.437394 | 37.721412 |

try to find some correlations between the target variable and the numeric variables but before that first remove the resolution column and describe the numeric columns to look for any missing values in the data.

In [106]:
```
train_cols = df.columns
train_cols
```

Out[106]:
```
Index(['Dates', 'Category', 'Descript', 'DayOfWeek', 'PdDistrict',
       'Resolution', 'Address', 'X', 'Y'],
      dtype='object')
```

In [107]:
```
test_cols = df_test.columns
test_cols
```

Out[107]:
```
Index(['Id', 'Dates', 'DayOfWeek', 'PdDistrict', 'Address', 'X', 'Y'], dtype='object')
```

In [108]:
```
cols = train_cols.drop("Resolution")
cols
```

Out[108]:
```
Index(['Dates', 'Category', 'Descript', 'DayOfWeek', 'PdDistrict', 'Address',
       'X', 'Y'],
      dtype='object')
```

In [109]:
```
train_new = df[cols]
train_new.head()
```

Out[109]:

| | Dates | Category | Descript | DayOfWeek | PdDistrict | Address | X | Y |
|---|---|---|---|---|---|---|---|---|
| **0** | 2015-05-13 23:53:00 | 1 | WARRANT ARREST | 3 | 1 | OAK ST / LAGUNA ST | -122.425892 | 37.774599 |
| **1** | 2015-05-13 23:53:00 | 2 | TRAFFIC VIOLATION ARREST | 3 | 1 | OAK ST / LAGUNA ST | -122.425892 | 37.774599 |
| **2** | 2015-05-13 23:33:00 | 2 | TRAFFIC VIOLATION ARREST | 3 | 1 | VANNESS AV / GREENWICH ST | -122.424363 | 37.800414 |
| **3** | 2015-05-13 23:30:00 | 3 | GRAND THEFT FROM LOCKED AUTO | 3 | 1 | 1500 Block of LOMBARD ST | -122.426995 | 37.800873 |
| **4** | 2015-05-13 23:30:00 | 3 | GRAND THEFT FROM LOCKED AUTO | 3 | 2 | 100 Block of BRODERICK ST | -122.438738 | 37.771541 |

In [110]:
```
train_new.describe()
```

Out[110]:

| | Category | DayOfWeek | PdDistrict | X | Y |
|---|---|---|---|---|---|
| **count** | 878049.000000 | 878049.000000 | 878049.000000 | 878049.000000 | 878049.000000 |
| **mean** | 7.224975 | 3.992691 | 6.037957 | -122.422616 | 37.771020 |
| **std** | 6.111544 | 1.972023 | 3.114945 | 0.030354 | 0.456893 |
| **min** | 1.000000 | 1.000000 | 1.000000 | -122.513642 | 37.707879 |
| **25%** | 3.000000 | 2.000000 | 3.000000 | -122.432952 | 37.752427 |
| **50%** | 5.000000 | 4.000000 | 6.000000 | -122.416420 | 37.775421 |
| **75%** | 10.000000 | 6.000000 | 9.000000 | -122.406959 | 37.784369 |
| **max** | 39.000000 | 7.000000 | 10.000000 | -120.500000 | 90.000000 |

In [111]:

```
# Finding Correlation of Output column with other column
corr = train_new.corr()
corr["Category"]
# There is no strong correlation of Category with other
```

Out[111]:

```
Category      1.000000
DayOfWeek    -0.016263
PdDistrict    0.007643
X             0.000147
Y            -0.005303
Name: Category, dtype: float64
```

In [112]:

```
#Calculate the skew

skew = train_new.skew()
skew
```

Out[112]:

```
Category        1.662607
DayOfWeek      -0.005572
PdDistrict     -0.232137
X              18.685494
Y             113.984988
dtype: float64
```

# Applying KNN model

In [144]:

```
#Let's use knn algorithm on numeric columns

features = ["DayOfWeek", "PdDistrict",  "X", "Y"]
X_train = train_new[features]
y_train = train_new["Category"]
X_test = df_test[features]
```

In [114]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [115]:

```
knn = KNeighborsClassifier(n_neighbors=1)
```

In [116]:

```
knn.fit(X_train, y_train)
```

Out[116]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=1, p=2,
           weights='uniform')
```

In [117]:

```
features
```

Out[117]:

```
['DayOfWeek', 'PdDistrict', 'X', 'Y']
```

In [118]:
```
X_test.head()
```

Out[118]:

| | DayOfWeek | PdDistrict | X | Y |
|---|---|---|---|---|
| **0** | 7 | 1 | -122.399588 | 37.735051 |
| **1** | 7 | 1 | -122.391523 | 37.732432 |
| **2** | 7 | 2 | -122.426002 | 37.792212 |
| **3** | 7 | 3 | -122.437394 | 37.721412 |
| **4** | 7 | 3 | -122.437394 | 37.721412 |

In [119]:
```
pred = knn.predict(X_test)
```

# Predictions and Evaluations

Let's evaluate our KNN model!

In [151]:
```
from collections import OrderedDict
data_dict_new = OrderedDict(sorted(data_dict.items()))
print(data_dict_new)
```

```
OrderedDict([('ARSON', 26), ('ASSAULT', 8), ('BAD CHECKS', 36), ('BRIBERY', 29), ('BURGLARY', 10), ('DISORDERLY CONDUCT', 25), ('DRIVING UNDER THE INFLUENCE', 22), ('DRUG/NARCOTIC', 14),
('DRUNKENNESS', 12), ('EMBEZZLEMENT', 30), ('EXTORTION', 34), ('FAMILY OFFENSES', 27), ('FORGERY/COUNTERFEITING', 13), ('FRAUD', 19), ('GAMBLING', 35), ('KIDNAPPING', 20), ('LARCENY/THEF
T', 3), ('LIQUOR LAWS', 28), ('LOITERING', 32), ('MISSING PERSON', 18), ('NON-CRIMINAL', 6), ('OTHER OFFENSES', 2), ('PORNOGRAPHY/OBSCENE MAT', 39), ('PROSTITUTION', 24), ('RECOVERED VEHI
CLE', 38), ('ROBBERY', 7), ('RUNAWAY', 21), ('SECONDARY CODES', 16), ('SEX OFFENSES FORCIBLE', 23), ('SEX OFFENSES NON FORCIBLE', 33), ('STOLEN PROPERTY', 15), ('SUICIDE', 31), ('SUSPICIO
US OCC', 11), ('TREA', 37), ('TRESPASS', 17), ('VANDALISM', 5), ('VEHICLE THEFT', 4), ('WARRANTS', 1), ('WEAPON LAWS', 9)])
```

In [155]:
```
#print type prediction

result = pd.DataFrame({"Id": df_test["Id"]})
for key,value in data_dict_new.items():
    result[key] = 0
count = 0
for item in pred:
    for key,value in data_dict.items():
        if value == item:
            result[key][count] = 1
    count += 1
result.to_csv("submission_knn.csv", index=False)
```

In [157]:
```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
lgr = LogisticRegression()
lgr.fit(X_train, y_train)
```

Out[157]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [158]:
```
pred = knn.predict(X_test)
```

In [161]:

```
#print(type(predictions))
result = pd.DataFrame({"Id": df_test["Id"]})
for key,value in data_dict_new.items():
    result[key] = 0
count = 0
for item in pred:
    for key,value in data_dict.items():
        if(value == item):
            result[key][count] = 1
    count+=1
result.to_csv("submission_logistic.csv", index=False)
```

In [ ]:

```
from sklearn.linear_model import LogisticRegression
log = LogisticRegression()
log.fit(X_train, y_train)
pred = log.predict(X_test)
```

In [ ]:

```
for key,value in data_dict_new.items():
    result_dataframe[key] = 0
count = 0
for item in predictions:
    for key,value in data_dict.items():
        if(value == item):
            result_dataframe[key][count] = 1
    count+=1
result.to_csv("submission_logistic.csv", index=False)
```