# Python - K-Nearest_Neighbors(KNN) model Cheat Sheet

by Pitbull (aggialavura) via cheatography.com/83764/cs/19947/

## TO START

```
# IMPORT DATA LIBRARIES
import numpy as np
import pandas as pd
# IMPORT VIS LIBRARIES
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
# IMPORT MODELLING LIBRARIES
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_tes-
t_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_rep-
ort,confusion_matrix
```

## PRELIMINARY OPERATIONS

df = pd.read_csv('data.csv') read data

**STANDARDISE THE VARIABLES**

```
scaler = StandardScaler()
scaler.fit(df.drop('y',axis=1))
scaled_feat = scaler.transform(df.drop('y',axis=1))
df_new=pd.DataFrame(scaled_feat,columns=df.columns[:-1])*
```

**df.columns[:-1]**: means take all the columns but the last one.

## TRAIN MODEL

**CREATE X and y**

X = df[['col1','col2',etc.]] create df features

y = df['col'] create df var to predict

**SPLIT DATASET**

```
X_train, X_test, y_train, y_test =
train_test_split(
  X,
  y,
  test_size=0.3)
```

split df in train and test df

**FIT THE MODEL**

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

knn.fit(X_train,y_train) train/fit the model

**MAKE PREDICTIONS**

## TRAIN MODEL (cont)

pred = knn.predict(X_test) make predictions

**n_neighbors=1**: we start specifying K = 1 and then we see how to better choose the K value (see evaluate block in this cheat sheet).

## EVALUATION of the MODEL

**EVAUATE MODEL**

```
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

**CHOOSING BETTER K**

error_rate = []* create an empty list

```
for i in range(1,40):
 knn = KNeighborsClassifier(n_neighbors=i)
 knn.fit(X_train,y_train)
 pred_i = knn.predict(X_test)
 error_rate.append(np.mean(pred_i != y_test))
```

**ELBOW PLOT**

```
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

Now we choose the K value where the error starts to reduce and flatten and we repeat the model fitting and evaluation! Theoretically, you should obtain better results.

**Explanation**:

1. we create an empty list.
2. we loop for a certain range of possible K values, here 1 to 40.
3. we create and fit the KNN model with these different K values.
4. we predict the using these models
5. we calculate the mean of the error of all these models and store the errors in the empty list of point 1. We will then plot these errors to see what K values could be the best one.