Importing/Uploading Data

In [ ]:
```
import pandas as pd
import numpy as np

data = pd.read_csv('/Users/juanrquilesjr/Downloads/Data_Mining_2019/DM_Project.csv')
pd.set_option('display.max_columns', 45)
pd.set_option('display.max_rows', 320)

data = data.drop(data.index[[310, 311, 312]])
data.head()
```

Statistics

In [ ]:
```
data.groupby('TJ Sx').mean()
```

KNN

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

import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
sns.set()

#Feature Selection
X = data[['4-Seam','Slider', 'Curve', 'Sinker', 'Cutter']].values
y = data['TJ Sx'].values

k_range = range(1,21)

Accuracy_scores = []

for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X, y, cv=10, scoring='accuracy')
    Accuracy_scores.append(scores.mean())
print(max(Accuracy_scores))
```

Logistic Regression

In [ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics

logreg = LogisticRegression(penalty = 'l2', solver = 'saga', max_iter = 50000, to
l = 1e-6)


#Feature Selection
X = data[['4-Seam','Slider', 'Curve', 'Sinker', 'Cutter']].values
y = data['TJ Sx'].values


CV_scores = cross_val_score(logreg, X, y, cv=10, scoring='accuracy')
print("CV_Scores:", cross_val_score(logreg, X, y, cv=10, scoring='accuracy'))
#print('Best CV Score:', max(cross_val_score(logreg, X, y, cv=10, scoring='accura
cy')))
print('Mean CV Scores:',CV_scores.mean())
```

Naive Bayes

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

#Feature Selection
X = data[['4-Seam','Slider', 'Curve', 'Sinker', 'Cutter']].values
y = data['TJ Sx'].values

nb = GaussianNB()
scores = cross_val_score(nb, X, y, cv=10, scoring='accuracy')
print(scores.mean())
```

SVM

In [ ]:
```
from sklearn import svm


#Feature Selection
X = data[['Cutter']].values
y = data['TJ Sx'].values

Create a svm Classifier
svm = svm.SVC(kernel='linear', gamma = 3, C = 2)

print(scores.mean())
```

Neural Network

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

#Feature Selection
X = data[['4-Seam','Slider', 'Curve', 'Sinker', 'Cutter', '# of Pitch Type']].val
ues
y = data['TJ Sx'].values

nodes = [25, 50, 75, 100, 125, 150, 175,200,250]

Accuracy_scores = []

for n in nodes:
    mlp = MLPClassifier(solver='sgd', hidden_layer_sizes=(n), random_state=1, max
_iter = 5000)
    scores = cross_val_score(mlp, X, y, cv=10, scoring='accuracy')
    Accuracy_scores.append(scores.mean())
print(Accuracy_scores)
print("Best Accuracty Mean Score:", max(Accuracy_scores))
```

Deep Learning

In [ ]:

```
from keras.layers import Dense
from keras.models import Sequential
from keras.utils import to_categorical

#Feature Selection
predictors = data[['4-Seam','Slider', 'Curve', 'Sinker', 'Cutter']].values
target = to_categorical(data['TJ Sx'].values)


n_cols = 5

model = Sequential()

model.add(Dense(150, activation='relu', input_shape=(n_cols,)))
model.add(Dense(150, activation = 'relu'))
model.add(Dense(150, activation = 'relu'))
model.add(Dense(150, activation = 'relu'))
model.add(Dense(150, activation = 'relu'))

model.add(Dense(2, activation = 'softmax'))

model.compile(optimizer = 'sgd', loss = 'categorical_crossentropy', metrics = ['a
ccuracy'])

model.fit(predictors, target, epochs = 300, batch_size = 20)

import matplotlib.pyplot as plt
%matplotlib inline

import numpy as np

batch = [1,5,10,15, 20]
scores = [.7645, .8710, .9226, .9548, .9645]

plt.plot(batch, scores)
plt.title('Deep Learning Scores w/ Different Batch Sizes & SGD Optimizer')
plt.xlabel('Batch Sizes')
plt.xticks(np.arange(0, 21, 5))
plt.ylabel('Accuracy Scores')
```