Problem Set 4
Noah Buchanan
April 8th, 2021

April 8, 2021

I will first show the best accuracy and hyper parameters after experimentation was done on each dataset. (Best Results Table)

Student Data
Best Accuracies

	Random Forest		Decision Tree		SVM		KNN
	Max Features of 6		Max Features of 7		Poly		distance weight function
Max Depth of 8	0.9494	Max Depth of 3	0.9241	C value above 1.2	0.9578	8 neighbors	0.9241

Cars
Best Accuracies

	Random Forest		Decision Tree		SVM		KNN
	Max Features of 5		Max Features of 10		Linear		distance weight function
Max Depth of 2	1	Max Depth of ANY	0.9524	C value above 0.2	0.9048	above 4 neighbors	0.9524

Titanic
Best Accuracies

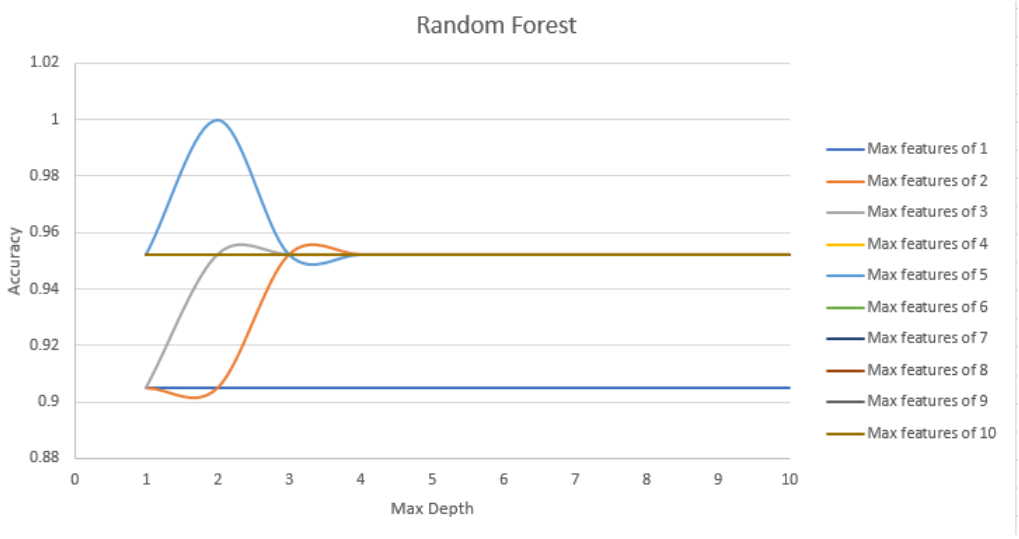
	Random Forest		Decision Tree		SVM		KNN
	Max Features of 3		Max Features of 3		Linear		distance weight function
Max Depth of 3	0.7786	Max Depth of 3	0.7786	ANY C value	0.7786	3 neighbors	0.7739

Experimentation of all datasets and classifier hyper parameters were done using a nested loop and at each iteration of each loop a parameter would change finding all permutations of hyper parameters and their accuracy. This was done 3 times each with a different 80 20 split, however all classifiers were tested once BEFORE splitting the data again to ensure equality among classifiers. The average of the 3 runs was taken for the final results you see.

1 Cars dataset

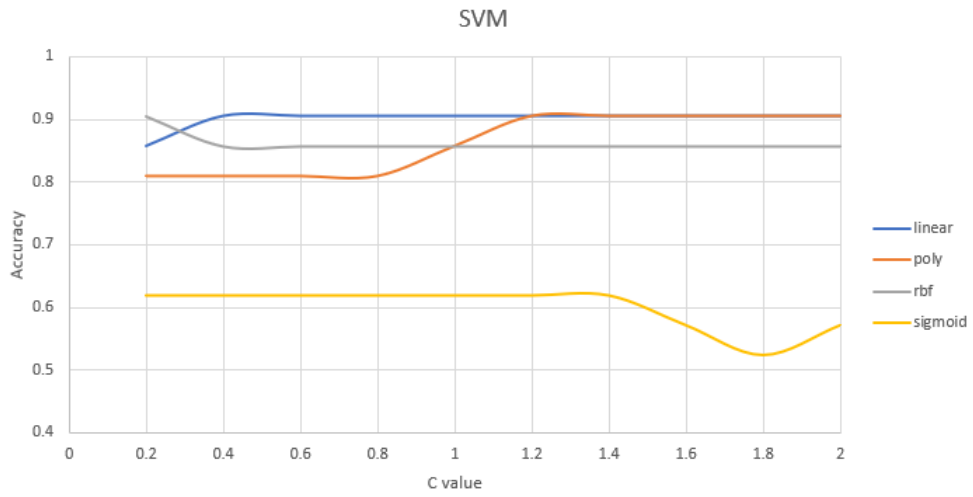
Slight modifications were made to this dataset to accomodate only numerical categorical values, converted all continuous features to 1 and 0 or hi and lo values.

1.1 Random Forest

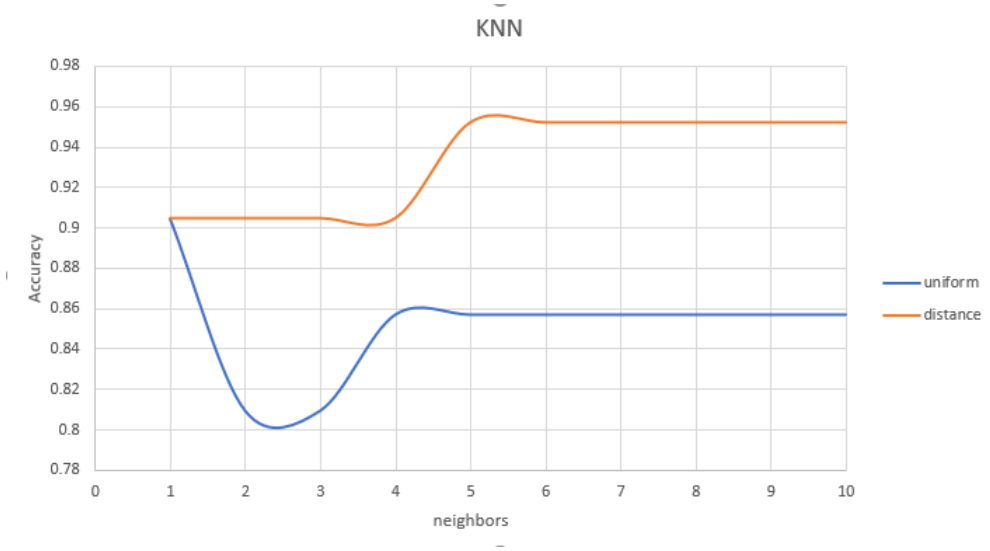


1.2 Decision Tree

1.3 Support Vector Machine



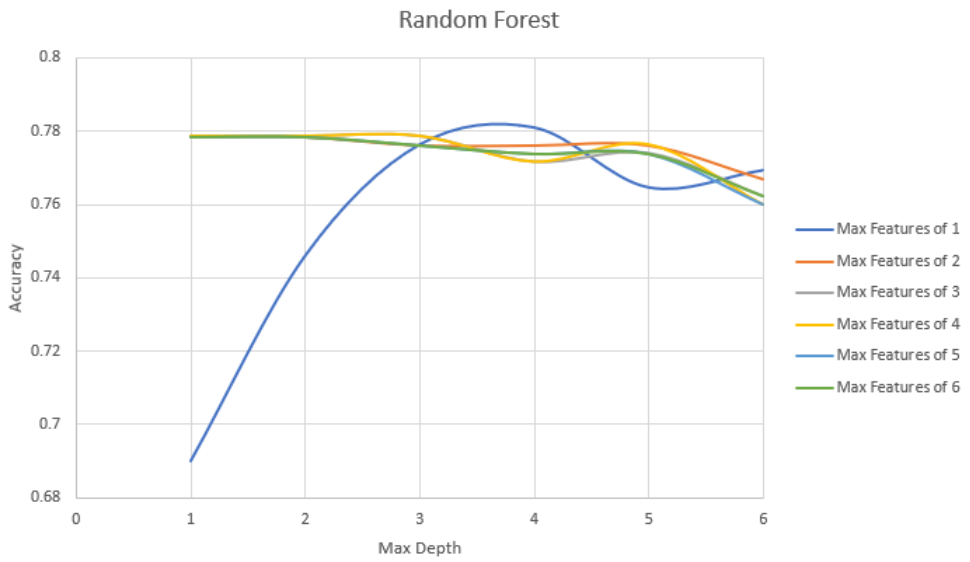
1.4 K-Nearest Neighbor



2 Titanic dataset

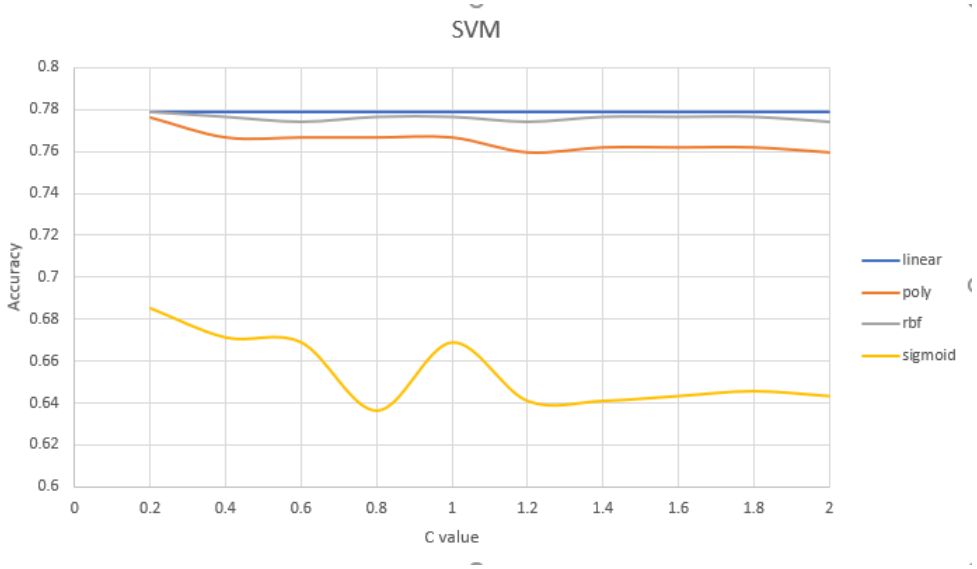
There were no data modifications for this dataset everything ran as is.

2.1 Random Forest

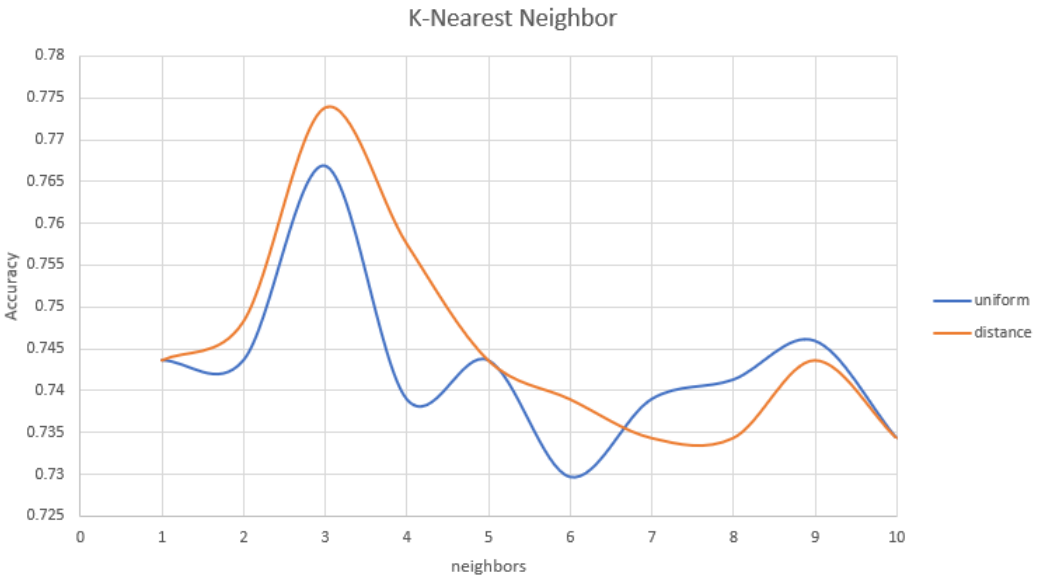


2.2 Decision Tree

2.3 Support Vector Machine



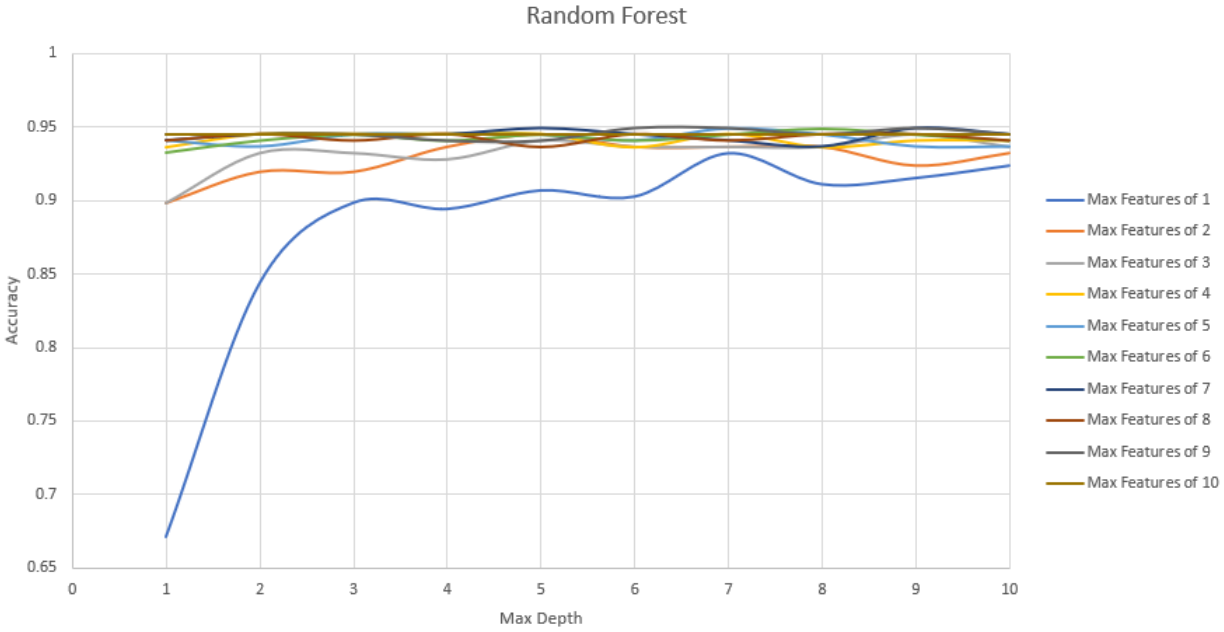
2.4 K-Nearest Neighbor



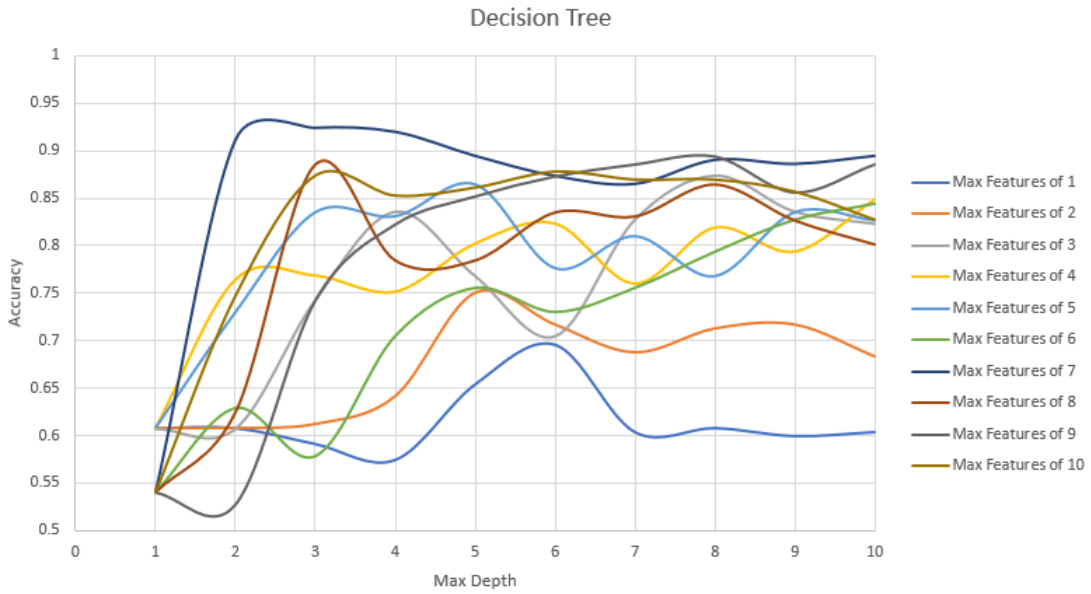
3 Student Data dataset

Modifications were made to accomodate only numerical categorical values for features and target column was converted to a hi lo range of 1 and 0 if above or below mean.

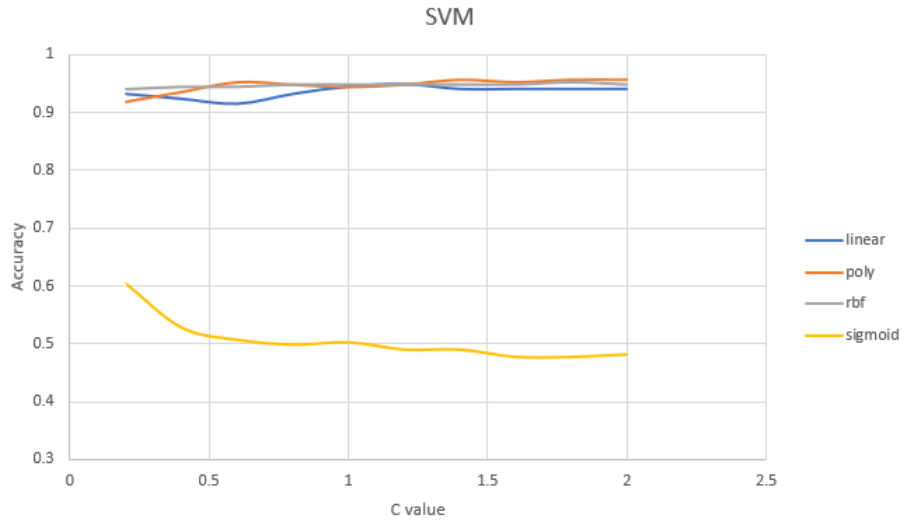
3.1 Random Forest



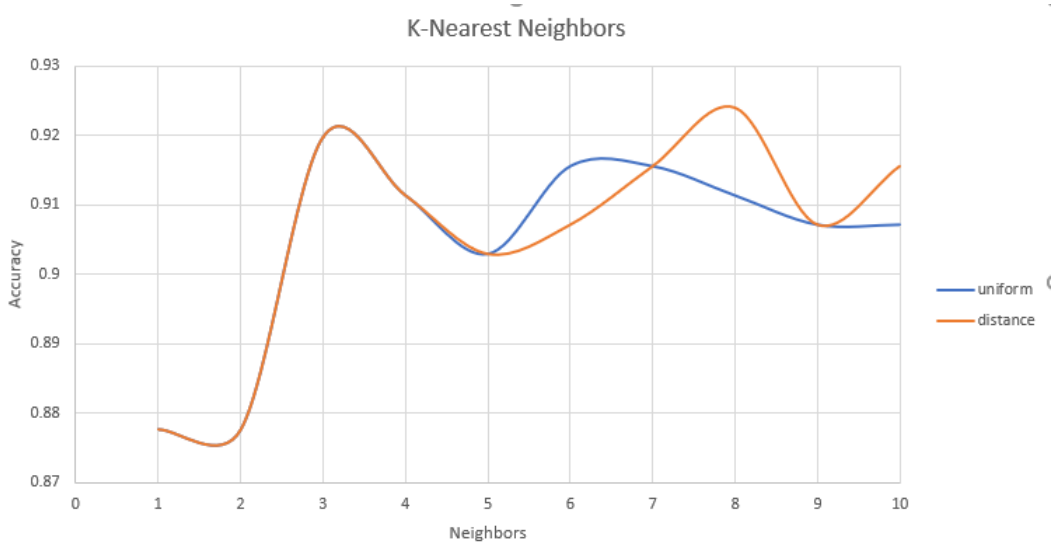
3.2 Decision Tree



3.3 Support Vector Machine



3.4 K-Nearest Neighbor



4 Cars Source Code

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd
import numpy as np

data = pd.read_csv("Cars.txt")

X = np.array(data.drop("mpg", axis=1)).astype(float)
y = np.array(data.pop("mpg")).astype(float)

sum = 0
for i in range(len(y)):
    sum += y[i]

avg = sum/len(y)

for i in range(len(y)):
    if y[i] > avg:
        y[i] = 1
```

```
    else:
        y[i] = 0

avgs = []
for i in range(1,6):
    sum = 0
    for j in range (len(X)):
        sum += X[j][i]
    avgs.append(sum/len(X))

for i in range(1,6):
    for j in range (len(X)):
        if X[j][i] > avgs[i-1]:
            X[j][i] = 1
        else:
            X[j][i] = 0

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
y_test = np.array(y_test).astype(int)


for i in range(1,11):
    for j in range(1,11):
        rf = RandomForestClassifier(max_features=i, max_depth=j, random_state=0)
        rf.fit(X_train,y_train)
        yhat = rf.predict(X_test)
        correct = 0;
        for h in range(len(yhat)):
            if y_test[h] == yhat[h]:
                correct += 1

        print(correct/len(yhat))
    print(",")


for i in range(1,11):
    for j in range(1,11):
        dt = DecisionTreeClassifier(max_features = i, max_depth = j,random_state
        dt.fit(X_train,y_train)
        yhat = dt.predict(X_test)
        correct = 0;
        for h in range(len(yhat)):
            if y_test[h] == yhat[h]:
                correct += 1

        print(correct/len(yhat))
```

```
    print(",")


kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for i in range(len(kernels)):
    for j in range(1,11):
        svc = svm.SVC(kernel = kernels[i],C = j*.2)
        svc.fit(X_train, y_train)
        yhat = svc.predict(X_test)
        correct = 0;
        for h in range(len(yhat)):
            if y_test[h] == yhat[h]:
                correct += 1

        print(correct/len(yhat))
    print(",")



weights = ['uniform', 'distance']
for i in range(1,11):
    for j in range(2):
        neigh = KNeighborsClassifier(n_neighbors=i, weights=weights[j])
        neigh.fit(X_train, y_train)
        yhat = neigh.predict(X_test)
        correct = 0;
        for h in range(len(yhat)):
            if y_test[h] == yhat[h]:
                correct += 1

        print(correct/len(yhat))
    print(",")
```

5 Titanic Source Code

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd
import numpy as np

data = pd.read_csv("Titanic.txt")
```

```
X = data.drop("SURVIVED", axis=1)
y = data.pop("SURVIVED")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
y_test = np.array(y_test).astype(int)

for i in range(1,7):
    for j in range(1,7):
        rf = RandomForestClassifier(max_features=i, max_depth=j, random_state=0)
        rf.fit(X_train,y_train)
        yhat = rf.predict(X_test)
        correct = 0;
        for h in range(len(yhat)):
            if y_test[h] == yhat[h]:
                correct += 1

        print(correct/len(yhat))
    print(",")

for i in range(1,7):
    for j in range(1,7):
        dt = DecisionTreeClassifier(max_features = i, max_depth = j,random_state
        dt.fit(X_train,y_train)
        yhat = dt.predict(X_test)
        correct = 0;
        for h in range(len(yhat)):
            if y_test[h] == yhat[h]:
                correct += 1

        print(correct/len(yhat))
    print(",")

kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for i in range(len(kernels)):
    for j in range(1,11):
        svc = svm.SVC(kernel = kernels[i],C = j*.2)
        svc.fit(X_train, y_train)
        yhat = svc.predict(X_test)
        correct = 0;
        for h in range(len(yhat)):
            if y_test[h] == yhat[h]:
                correct += 1

        print(correct/len(yhat))
    print(",")

weights = ['uniform', 'distance']
```

```
for i in range(1,11):
    for j in range(2):
        neigh = KNeighborsClassifier(n_neighbors=i, weights=weights[j])
        neigh.fit(X_train, y_train)
        yhat = neigh.predict(X_test)
        correct = 0;
        for h in range(len(yhat)):
            if y_test[h] == yhat[h]:
                correct += 1

        print(correct/len(yhat))
    print(",")
```

6 Student Data Source Code

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd
import numpy as np

def convertCategorical(column):
    map_convert = get_mapping(column)
    keys.append(map_convert)

    for i in range(len(data[column])):
        data[column][i] = convert(data[column][i], map_convert)


def get_mapping(column):
    map_convert = {}
    unique_values = data[column].tolist()
    unique_values = set(unique_values)
    unique_values = list(unique_values)
    count = 0

    for value in unique_values:
        map_convert[value] = count
        count += 1

    return map_convert
```

```
def convert(value, map):
    return map[value]


global keys
keys = list()
global data
data = pd.read_csv("StudentData.txt")

for column in data.columns:
    convertCategorical(column)

X = data.drop("G3", axis=1)
y = np.array(data.pop("G3")).astype(float)

sum = 0
for i in range(len(y)):
    sum += y[i]

avg = sum/len(y)

for i in range(len(y)):
    if y[i] > avg:
        y[i] = 1
    else:
        y[i] = 0


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
y_test = np.array(y_test).astype(int)



for i in range(1,11):
    for j in range(1,11):
        rf = RandomForestClassifier(max_features=i, max_depth=j, random_state=0)
        rf.fit(X_train,y_train)
        yhat = rf.predict(X_test)
        correct = 0;
        for h in range(len(yhat)):
            if y_test[h] == yhat[h]:
                correct += 1

        print(correct/len(yhat))
    print(",")
```

```
for i in range(1,11):
    for j in range(1,11):
        dt = DecisionTreeClassifier(max_features = i, max_depth = j,random_state
        dt.fit(X_train,y_train)
        yhat = dt.predict(X_test)
        correct = 0;
        for h in range(len(yhat)):
            if y_test[h] == yhat[h]:
                correct += 1

        print(correct/len(yhat))
    print(",")



kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for i in range(len(kernels)):
    for j in range(1,11):
        svc = svm.SVC(kernel = kernels[i],C = j*.2)
        svc.fit(X_train, y_train)
        yhat = svc.predict(X_test)
        correct = 0;
        for h in range(len(yhat)):
            if y_test[h] == yhat[h]:
                correct += 1

        print(correct/len(yhat))
    print(",")


weights = ['uniform', 'distance']
for i in range(1,11):
    for j in range(2):
        neigh = KNeighborsClassifier(n_neighbors=i, weights=weights[j])
        neigh.fit(X_train, y_train)
        yhat = neigh.predict(X_test)
        correct = 0;
        for h in range(len(yhat)):
            if y_test[h] == yhat[h]:
                correct += 1

        print(correct/len(yhat))
    print(",")
```