In this notebook I will be attempting to predict how gender can affect math scores in tests, utilizing a data set from Kaggle.com on student test scores.

Loading libraries and dataset

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import graphviz

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import export_graphviz
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LinearRegression

from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsRegressor

import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
df = pd.read_csv("StudentsPerformance.csv")
df.head()
```

Out[2]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

In [3]:
```
df.dtypes
```

Out[3]:
```
gender                         object
race/ethnicity                 object
parental level of education    object
lunch                          object
test preparation course        object
math score                      int64
reading score                   int64
writing score                   int64
dtype: object
```

Decision Tree Regression

In [4]:
```
df2 = pd.get_dummies(df, columns = ["gender", "race/ethnicity", "parental leve
l of education", "lunch", "test preparation course"], drop_first = True)
df2.head()
```

Out[4]:

	math score	reading score	writing score	gender_male	race/ethnicity_group B	race/ethnicity_group C	race/
0	72	72	74	0	1	0	0
1	69	90	88	0	0	1	0
2	90	95	93	0	1	0	0
3	47	57	44	1	0	0	0
4	76	78	75	1	0	1	0

In [5]:
```
x = df2.drop(columns = ["math score"])
x.head()
```

Out[5]:

	reading score	writing score	gender_male	race/ethnicity_group B	race/ethnicity_group C	race/ethnici
0	72	74	0	1	0	0
1	90	88	0	0	1	0
2	95	93	0	1	0	0
3	57	44	1	0	0	0
4	78	75	1	0	1	0

In [6]:
```
y = df2["math score"]
y.head()
```

Out[6]:
```
0    72
1    69
2    90
3    47
4    76
Name: math score, dtype: int64
```

In [7]:
```
model = DecisionTreeRegressor(max_depth = 3)
model.fit(x, y)
```

Out[7]:
```
DecisionTreeRegressor(criterion='mse', max_depth=3, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [8]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

In [9]:
```
x_train.head()
```

Out[9]:

	reading score	writing score	gender_male	race/ethnicity_group B	race/ethnicity_group C	race/ethn
706	34	36	1	0	0	1
17	32	28	0	1	0	0
91	34	36	1	0	1	0
430	66	59	1	0	1	0
441	81	80	0	0	0	1

In [10]:
```
x_test.head()
```

Out[10]:

	reading score	writing score	gender_male	race/ethnicity_group B	race/ethnicity_group C	race/ethn
112	53	47	1	0	0	0
491	64	70	0	0	1	0
456	89	89	0	0	0	1
736	79	84	1	0	1	0
857	79	81	0	0	1	0

In [11]:
```
y_train.head()
```

Out[11]:
```
706    46
17     18
91     27
430    64
441    78
Name: math score, dtype: int64
```

In [12]:
```
y_test.head()
```

Out[12]:
```
112    54
491    64
456    79
736    92
857    65
Name: math score, dtype: int64
```

In [13]:
```
reg3 = DecisionTreeRegressor(max_depth = 3)
reg3 = reg3.fit(x_train, y_train)
```

In [14]:
```
predictions_3 = reg3.predict(x_test)
```

In [15]:
```
mean_squared_error(predictions_3, y_test)
```

Out[15]:
```
63.84479956102419
```

In [16]:
```
predictions_3_train = reg3.predict(x_train)
```

In [17]:
```
mean_squared_error(predictions_3_train, y_train)
```

Out[17]:
```
66.82417629543858
```

In [18]:
```
lin_model = LinearRegression()
lin_model.fit(x_train,y_train)
```

Out[18]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [19]:
```
y_test_preds_lin = lin_model.predict(x_test)
```

In [20]:
```
mean_squared_error(y_test_preds_lin, y_test)
```

Out[20]:
```
30.67924654739501
```

This decision tree is giving a small mean squared error, so it is doing a good job at predicting the data.

K-Nearest Neighbors Regression

In [21]:
```
knn = KNeighborsRegressor(n_neighbors = 7)
knn.fit(x_train, y_train)
```

Out[21]:
```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
          metric_params=None, n_jobs=None, n_neighbors=7, p=2,
          weights='uniform')
```

In [22]:
```
y_test_pred = knn.predict(x_test)
```

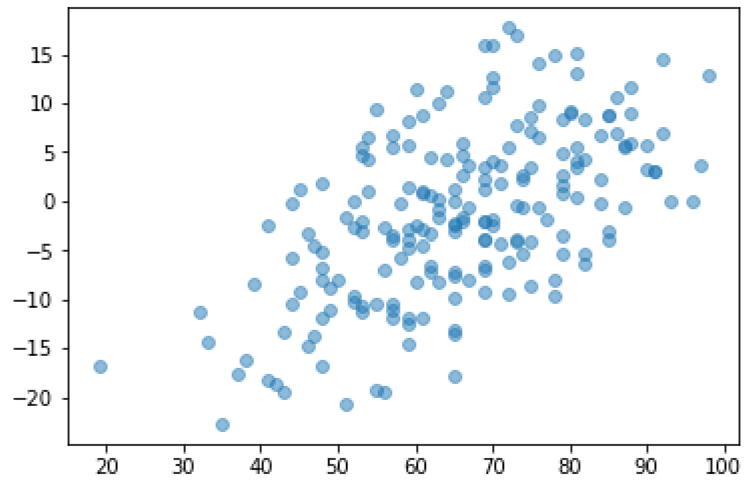
In [23]:
```
mean_squared_error(y_test_pred, y_test)
```

Out[23]:
```
74.16683673469387
```

In [24]:
```
y_train_pred = knn.predict(x_train)
```

In [25]:
```
plt.scatter(y_test, y_test - y_test_pred, alpha = 0.5)
```

Out[25]: <matplotlib.collections.PathCollection at 0x7f61695f8518>



In [26]:
```
scaler = MinMaxScaler()
```

In [27]:
```
x_train_scaled = scaler.fit_transform(x_train)
```

```
/usr/local/lib/python3.4/site-packages/sklearn/preprocessing/data.py:334: Dat
aConversionWarning: Data with input dtype uint8, int64 were all converted to
float64 by MinMaxScaler.
  return self.partial_fit(X, y)
```

In [28]:
```
x_train_scaled
```

Out[28]:
```
array([[0.20481928, 0.28888889, 1.        , ..., 0.        , 1.        ,
        1.        ],
       [0.18072289, 0.2       , 0.        , ..., 1.        , 0.        ,
        1.        ],
       [0.20481928, 0.28888889, 1.        , ..., 0.        , 0.        ,
        1.        ],
       ...,
       [0.73493976, 0.73333333, 0.        , ..., 0.        , 1.        ,
        0.        ],
       [0.54216867, 0.6       , 1.        , ..., 0.        , 0.        ,
        1.        ],
       [0.40963855, 0.45555556, 1.        , ..., 0.        , 0.        ,
        1.        ]])
```

In [29]:
```
x_test_scaled = scaler.fit_transform(x_test)
```

```
/usr/local/lib/python3.4/site-packages/sklearn/preprocessing/data.py:334: Dat
aConversionWarning: Data with input dtype uint8, int64 were all converted to
float64 by MinMaxScaler.
  return self.partial_fit(X, y)
```

In [30]:
```
knn_scaled = KNeighborsRegressor(n_neighbors = 7)
knn_scaled.fit(x_train_scaled, y_train)
```

Out[30]:
```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
          metric_params=None, n_jobs=None, n_neighbors=7, p=2,
          weights='uniform')
```

In [31]:
```
y_test_scaled_pred = knn_scaled.predict(x_test_scaled)
```

In [32]:
```
mean_squared_error(y_test_scaled_pred, y_test)
```

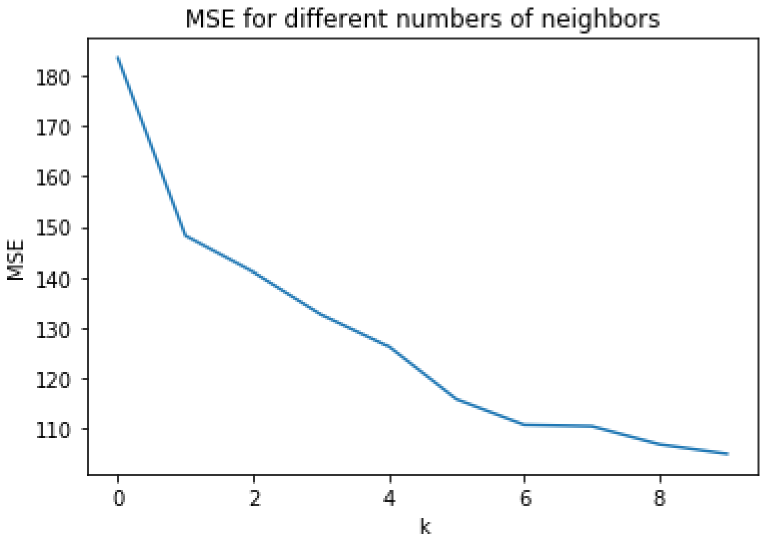
Out[32]:
```
110.76020408163264
```

In [33]:
```
mses = []
for k in range(1,11):
    print("Now computing MSE for k=",k)
    iknn_scaled = KNeighborsRegressor(n_neighbors = k)
    iknn_scaled.fit(x_train_scaled, y_train)
    iy_pred_scaled = iknn_scaled.predict(x_test_scaled)
    mse = mean_squared_error(iy_pred_scaled, y_test)
    mses.append(mse)
```

```
Now computing MSE for k= 1
Now computing MSE for k= 2
Now computing MSE for k= 3
Now computing MSE for k= 4
Now computing MSE for k= 5
Now computing MSE for k= 6
Now computing MSE for k= 7
Now computing MSE for k= 8
Now computing MSE for k= 9
Now computing MSE for k= 10
```

In [34]:
```
plt.plot(mses)
plt.xlabel("k")
plt.ylabel("MSE")
plt.title("MSE for different numbers of neighbors")
```

Out[34]: `Text(0.5,1,'MSE for different numbers of neighbors')`



Based on this graph, the MSE using this method doesn't justify the extra work and complexity of this method

---

When comparing the two prediction methods, the Decision Tree Regression worked better at predicting the data and was simpler than the K-Nearest Neighbors Regression method.