Project 3

1. Iris Dataset

1. A & C

In this problem we were suppose to solve k Nearest Neighbors aka KNN problem. First we were told to find the euclidean distance and we can find that by doing simple knn because it calculates the euclidean distance if it's set as default. This is a example of where k=1 problem and with the confusion matrix and accuracy report.

In [90]:
```
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df.head()
```

Out[90]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [91]:
```
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names )
df.head()
```

Out[91]:

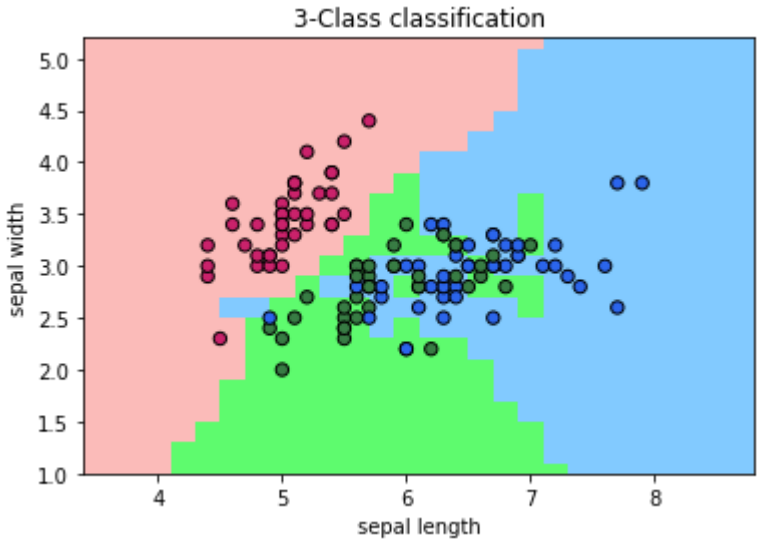
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

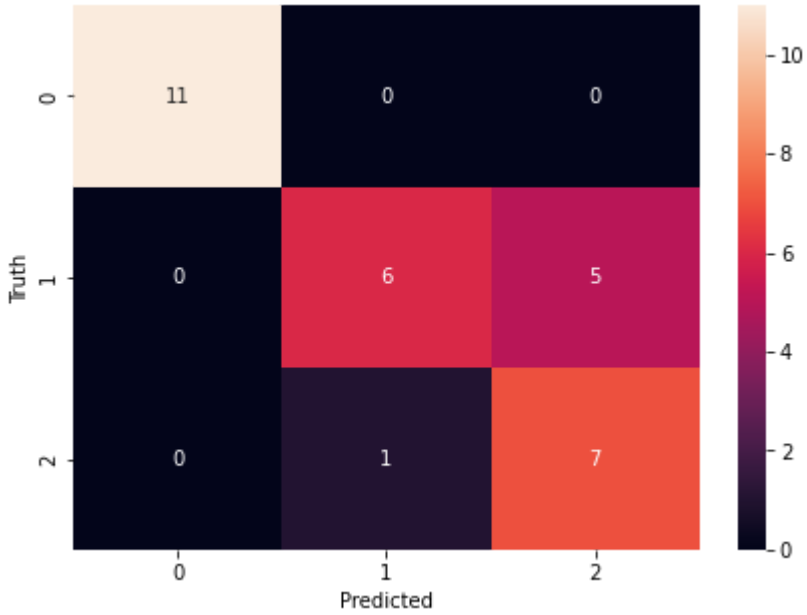
In [92]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(iris.data, iris.target, test_size = 0.
```

In [93]:
```
xtrain = X_train[:, :2]
xtest = X_test[:, :2]
```

In [94]:
```
%matplotlib inline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
```

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt1
from matplotlib.colors import ListedColormap
import seaborn as sn

cmap_light = ListedColormap(['#FBBBB9', '#5EFB6E', '#82CAFF'])
cmap_bold = ListedColormap(['#CA226B', '#387C44', '#2B65EC'])
cmap_test = ListedColormap(['#8E35EF', '#FFFF00', '#659EC7'])

#meshstep size parameter
h = 0.2

#KNN learner
model = KNeighborsClassifier(1)


#Fitting the data
model.fit(xtrain, y_train)
y_pred = model.predict(xtest)

# Plot the decision boundary
# For using meshgrid, you need to find the min max values of both attributes
# We usually make min/max a little lower/higher than the actual value
# here y is representing the second attributes, do not confuse it with the label
x_min, x_max = xtrain[:, 0].min() - 1, xtrain[:, 0].max() + 1
y_min, y_max = xtrain[:, 1].min() - 1, xtrain[:, 1].max() + 1


# make the meshgrid
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),np.arange(y_min, y_max, h))
# add the classifier to the meshgrid
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)


# plot the outcome
plt.figure()
plt.pcolormesh(xx, yy, Z, cmap=cmap_light, shading='auto')
plt.scatter(xtrain[:, 0], xtrain[:, 1], c=y_train, cmap=cmap_bold,edgecolor='k', s=40)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xlabel("sepal length")
plt.ylabel("sepal width")
plt.title("3-Class classification")
plt.show()

cm = confusion_matrix(y_test, y_pred)
plt1.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt1.xlabel('Predicted')
plt1.ylabel('Truth')
plt1.show()

from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```





	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	0.86	0.55	0.67	11
2	0.58	0.88	0.70	8
accuracy			0.80	30
macro avg	0.81	0.81	0.79	30
weighted avg	0.84	0.80	0.80	30

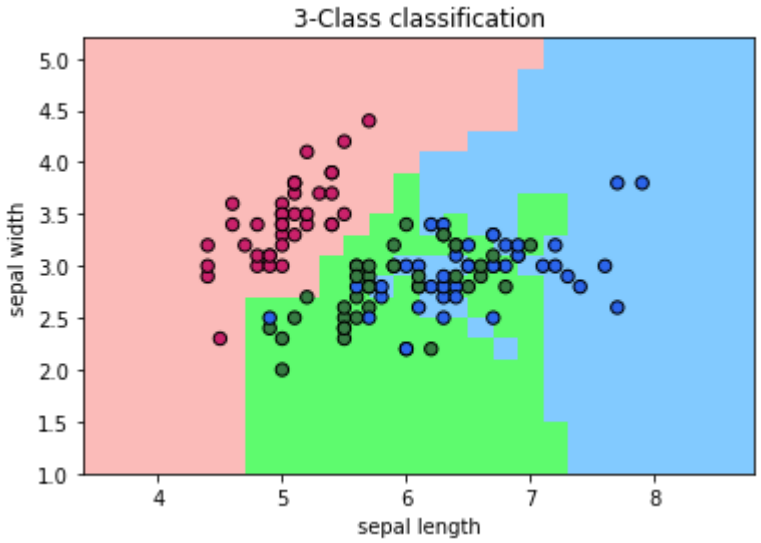
1. B & C

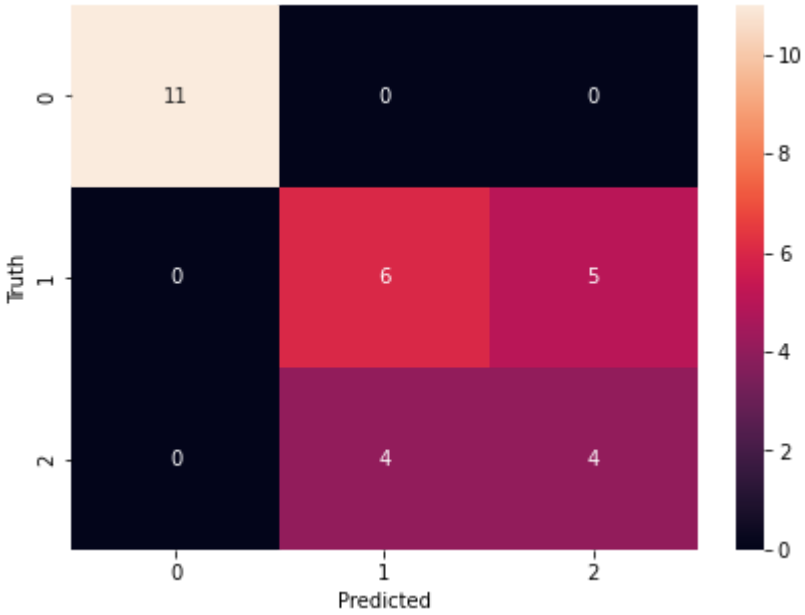
In this problem we were suppose to solve k Nearest Neighbors aka KNN problem. First we were told to find the euclidean distance and we can find that by doing simple knn because it calculates the euclidean distance if it's set as default. This is a example of where k=2, 4 and 6 problem and with the confusion matrix and accuracy report for all of the three k values.

In [95]:
```
xtrain = X_train[:, :2]
xtest = X_test[:, :2]
```

In [96]:
```
%matplotlib inline
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.metrics import confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt1
from matplotlib.colors import ListedColormap
import seaborn as sn

cmap_light = ListedColormap(['#FBBBB9', '#5EFB6E', '#82CAFF'])
cmap_bold = ListedColormap(['#CA226B', '#387C44', '#2B65EC'])
cmap_test = ListedColormap(['#8E35EF', '#FFFF00', '#659EC7'])

#meshstep size parameter
h = 0.2

#KNN learner
model = KNeighborsClassifier(2)


#Fitting the data
model.fit(xtrain, y_train)
y_pred = model.predict(xtest)

# Plot the decision boundary
# For using meshgrid, you need to find the min max values of both attributes
# We usually make min/max a little lower/higher than the actual value
# here y is representing the second attributes, do not confuse it with the label
x_min, x_max = xtrain[:, 0].min() - 1, xtrain[:, 0].max() + 1
y_min, y_max = xtrain[:, 1].min() - 1, xtrain[:, 1].max() + 1


# make the meshgrid
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),np.arange(y_min, y_max, h))
# add the classifier to the meshgrid
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)


# plot the outcome
plt.figure()
plt.pcolormesh(xx, yy, Z, cmap=cmap_light, shading='auto')
plt.scatter(xtrain[:, 0], xtrain[:, 1], c=y_train, cmap=cmap_bold,edgecolor='k', s=40)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xlabel("sepal length")
plt.ylabel("sepal width")
plt.title("3-Class classification")
plt.show()

cm = confusion_matrix(y_test, y_pred)
plt1.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt1.xlabel('Predicted')
plt1.ylabel('Truth')
plt1.show()

from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```





	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	0.60	0.55	0.57	11
2	0.44	0.50	0.47	8
accuracy			0.70	30
macro avg	0.68	0.68	0.68	30
weighted avg	0.71	0.70	0.70	30

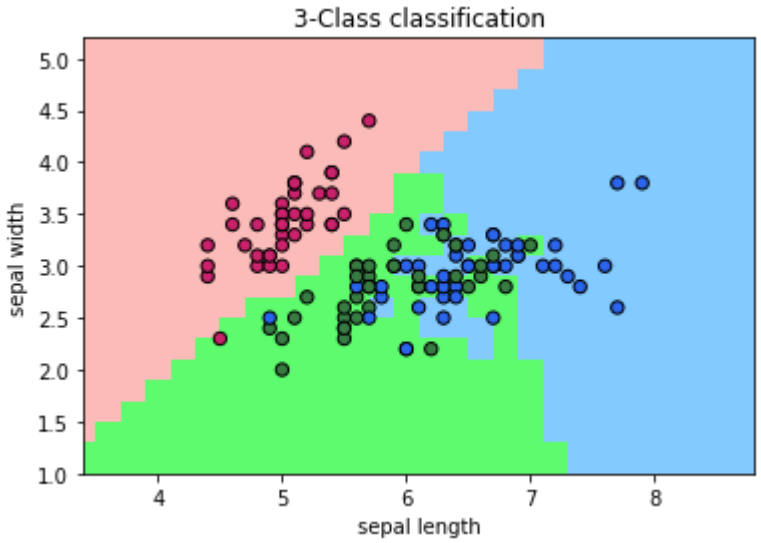
In [97]:

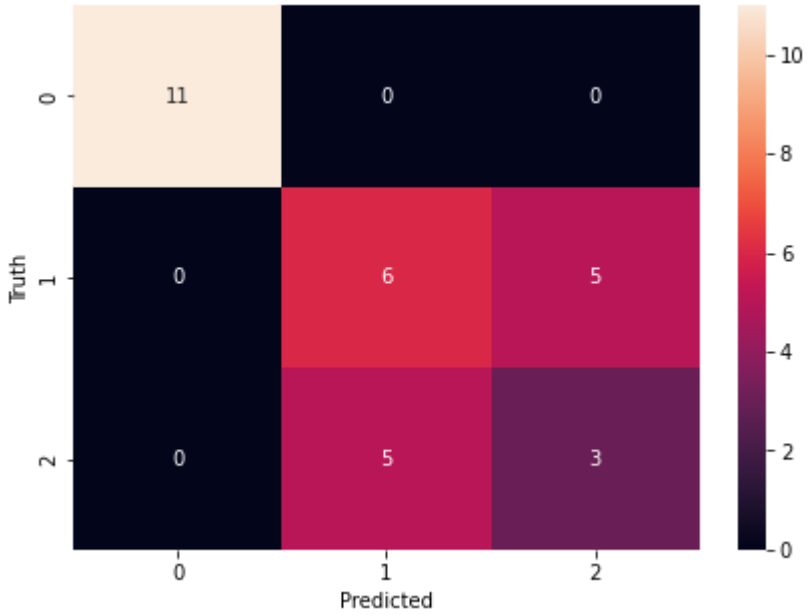
```
xtrain = X_train[:, :2]
xtest = X_test[:, :2]
```

In [98]:

```
%matplotlib inline
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

cmap_light = ListedColormap(['#FBBBB9', '#5EFB6E', '#82CAFF'])
```

```
cmap_bold = ListedColormap(['#CA226B', '#387C44', '#2B65EC'])
cmap_test = ListedColormap(['#8E35EF', '#FFFF00', '#659EC7'])

#meshstep size parameter
h = 0.2

#KNN learner
model = KNeighborsClassifier(4)


#Fitting the data
model.fit(xtrain, y_train)
y_pred = model.predict(xtest)

# Plot the decision boundary
# For using meshgrid, you need to find the min max values of both attributes
# We usually make min/max a little lower/higher than the actual value
# here y is representing the second attributes, do not confuse it with the label
x_min, x_max = xtrain[:, 0].min() - 1, xtrain[:, 0].max() + 1
y_min, y_max = xtrain[:, 1].min() - 1, xtrain[:, 1].max() + 1


# make the meshgrid
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),np.arange(y_min, y_max, h))
# add the classifier to the meshgrid
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)


# plot the outcome
plt.figure()
plt.pcolormesh(xx, yy, Z, cmap=cmap_light, shading='auto')
plt.scatter(xtrain[:, 0], xtrain[:, 1], c=y_train, cmap=cmap_bold,edgecolor='k', s=40)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xlabel("sepal length")
plt.ylabel("sepal width")
plt.title("3-Class classification")
plt.show()

cm = confusion_matrix(y_test, y_pred)
plt1.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt1.xlabel('Predicted')
plt1.ylabel('Truth')
plt1.show()

from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```



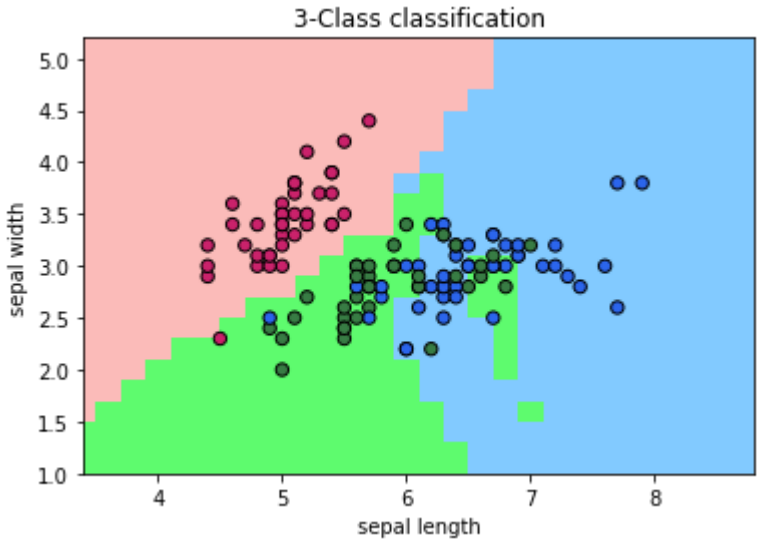


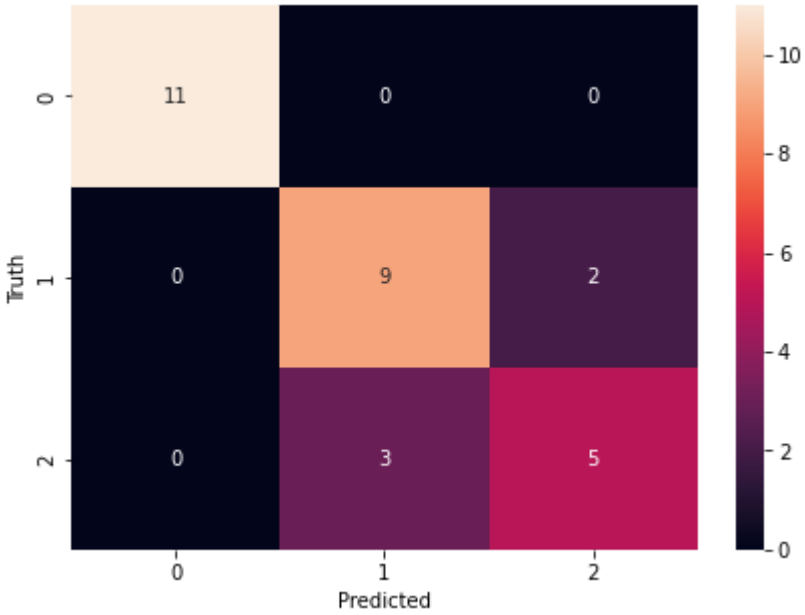
	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	0.55	0.55	0.55	11
2	0.38	0.38	0.38	8
accuracy			0.67	30
macro avg	0.64	0.64	0.64	30
weighted avg	0.67	0.67	0.67	30

In [99]:
```
xtrain = X_train[:, :2]
xtest = X_test[:, :2]
```

In [100…
```
%matplotlib inline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt1
import seaborn as sn
```

```
from matplotlib.colors import ListedColormap

cmap_light = ListedColormap(['#FBBBB9', '#5EFB6E', '#82CAFF'])
cmap_bold = ListedColormap(['#CA226B', '#387C44', '#2B65EC'])
cmap_test = ListedColormap(['#8E35EF', '#FFFF00', '#659EC7'])

#meshstep size parameter
h = 0.2

#KNN learner
model = KNeighborsClassifier(6)


#Fitting the data
model.fit(xtrain, y_train)
y_pred = model.predict(xtest)

# Plot the decision boundary
# For using meshgrid, you need to find the min max values of both attributes
# We usually make min/max a little lower/higher than the actual value
# here y is representing the second attributes, do not confuse it with the label
x_min, x_max = xtrain[:, 0].min() - 1, xtrain[:, 0].max() + 1
y_min, y_max = xtrain[:, 1].min() - 1, xtrain[:, 1].max() + 1


# make the meshgrid
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),np.arange(y_min, y_max, h))
# add the classifier to the meshgrid
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)


# plot the outcome
plt.figure()
plt.pcolormesh(xx, yy, Z, cmap=cmap_light, shading='auto')
plt.scatter(xtrain[:, 0], xtrain[:, 1], c=y_train, cmap=cmap_bold,edgecolor='k', s=40)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xlabel("sepal length")
plt.ylabel("sepal width")
plt.title("3-Class classification")
plt.show()

cm = confusion_matrix(y_test, y_pred)
plt1.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt1.xlabel('Predicted')
plt1.ylabel('Truth')
plt1.show()

from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```





	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	0.75	0.82	0.78	11
2	0.71	0.62	0.67	8
accuracy			0.83	30
macro avg	0.82	0.81	0.82	30
weighted avg	0.83	0.83	0.83	30

2. Bank notes Dataset

2. A

In this problem we were suppose to solve k Nearest Neighbors aka KNN problem. First we were told to find the euclidean distance and we can find that by doing simple knn because it calculates the euclidean distance if it's set as default. This is a example of where k=2 problem and with the confusion matrix and accuracy report .

In [101…

```
banknotes = pd.read_csv("data_banknote_authentication.csv", names=['Variance of Wavelet
```

In [102…

```
banknotes.head()
```

Out[102…

	Variance of Wavelet	Skewness of Wavelet	Kurtosis of Wavelet	Entropy of image	class
0	4.54590	8.1674	-2.4586	-1.46210	0
1	3.86600	-2.6383	1.9242	0.10645	0
2	3.45660	9.5228	-4.0112	-3.59440	0
3	0.32924	-4.4552	4.5718	-0.98880	0
4	4.36840	9.6718	-3.9606	-3.16250	0

In [103…

```
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import svm
from sklearn.linear_model import LogisticRegression
import pandas as pd
import scipy as sy
import os
import matplotlib.pyplot as plt

array = banknotes.values
X = array[:, 0:4]
y = array[:,4]
```

In [104…

```
print(X)
```

```
[[  4.5459    8.1674   -2.4586   -1.4621 ]
 [  3.866    -2.6383    1.9242    0.10645]
 [  3.4566    9.5228   -4.0112   -3.5944 ]
 ...
 [ -3.7503  -13.4586   17.5932   -2.7771 ]
 [ -3.5637   -8.3827   12.393    -1.2823 ]
 [ -2.5419   -0.65804   2.6842    1.1952 ]]
```

In [105…

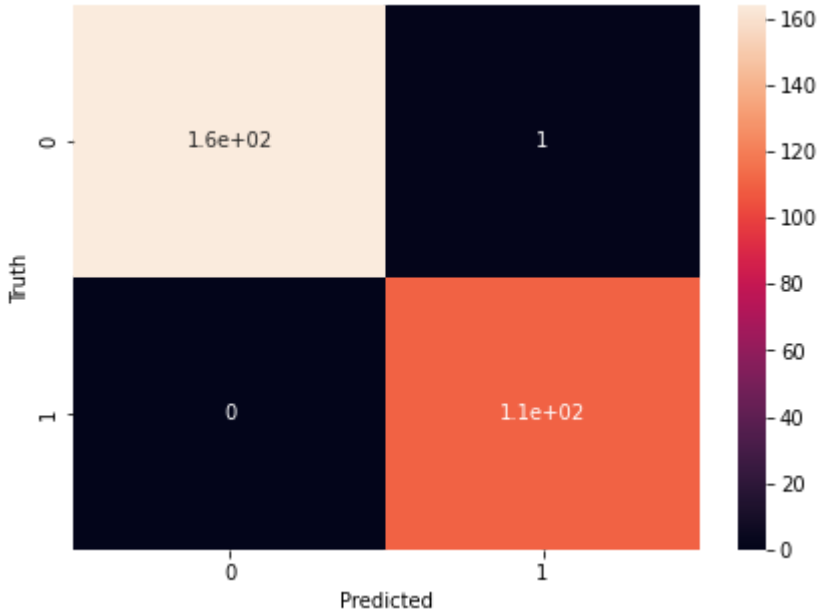
```
print(y)
```

```
[0. 0. 0. ... 1. 1. 1.]
```

In [124…

```
%matplotlib inline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt1
import seaborn as sn
from matplotlib.colors import ListedColormap

bank_X_train,bank_X_test,bank_y_train,bank_y_test = train_test_split(X, y, test_size =

#KNN Learner
model = KNeighborsClassifier(2)
```

```
#Fitting the data
model.fit(bank_X_train, bank_y_train)
y_pred = model.predict(bank_X_test)

cm = confusion_matrix(bank_y_test, y_pred)
plt1.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt1.xlabel('Predicted')
plt1.ylabel('Truth')
plt1.show()

from sklearn.metrics import classification_report
print(classification_report(bank_y_test,y_pred))
```




```
              precision    recall  f1-score   support

         0.0       1.00      0.99      1.00       165
         1.0       0.99      1.00      1.00       110

    accuracy                           1.00       275
   macro avg       1.00      1.00      1.00       275
weighted avg       1.00      1.00      1.00       275
```

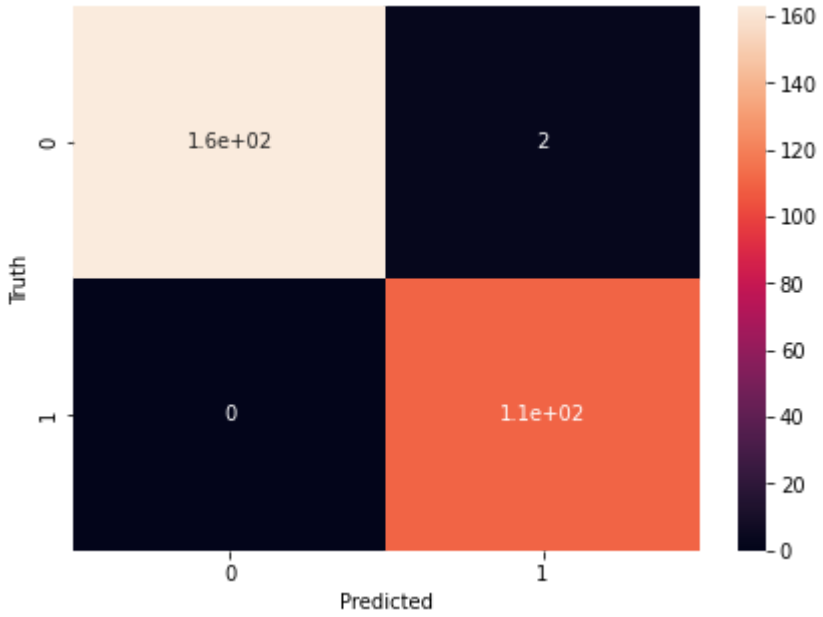
2. B & D

In this problem we were suppose to change the majority based voting choosen of yourself. And then describe the error rate and also the confusion matrix and the accurcy report.

In [125…

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(random_state=0)
clf.fit(bank_X_train, bank_y_train)
y2_pred = clf.predict(bank_X_test)
cm = confusion_matrix(bank_y_test, y2_pred)
plt1.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt1.xlabel('Predicted')
plt1.ylabel('Truth')
plt1.show()
```

```
from sklearn.metrics import classification_report
print(classification_report(bank_y_test,y2_pred))
```




```
              precision    recall  f1-score   support

         0.0       1.00      0.99      0.99       165
         1.0       0.98      1.00      0.99       110

    accuracy                           0.99       275
   macro avg       0.99      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275
```

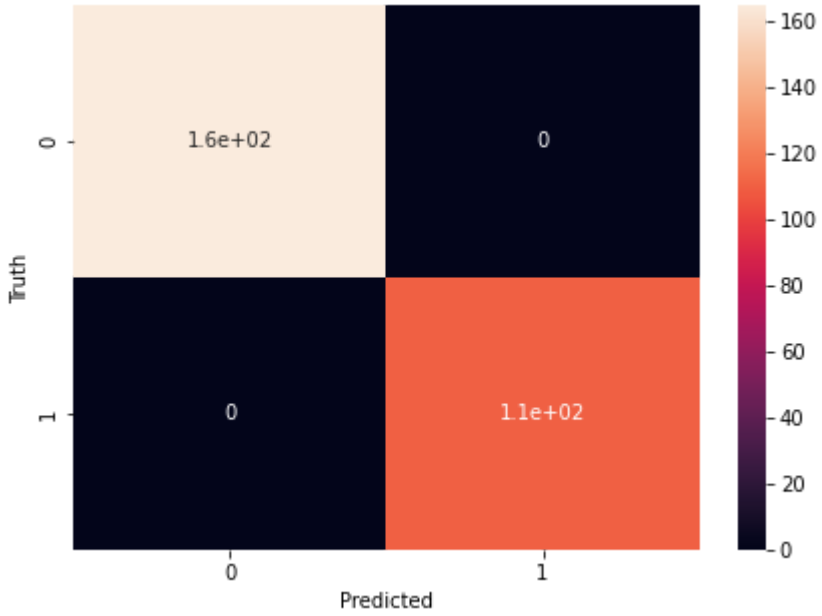
2. C & D

In this problem we were suppose to solve k Nearest Neighbors aka KNN problem. First we were told to find the manhattan distance and we can find that by doing knn with p=1 and everything other as default because it calculates the manhattan distance. This is a example of where k=2 problem and with the confusion matrix and accuracy report .

In [132…

```
%matplotlib inline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt1
import seaborn as sn
from matplotlib.colors import ListedColormap

#KNN Learner
model = KNeighborsClassifier(n_neighbors=2, p=1, metric='minkowski')


#Fitting the data
model.fit(bank_X_train, bank_y_train)
y_pred = model.predict(bank_X_test)

cm = confusion_matrix(bank_y_test, y_pred)
plt1.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
```

```
plt1.xlabel('Predicted')
plt1.ylabel('Truth')
plt1.show()

from sklearn.metrics import classification_report
print(classification_report(bank_y_test,y_pred))
```




```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       165
         1.0       1.00      1.00      1.00       110

    accuracy                           1.00       275
   macro avg       1.00      1.00      1.00       275
weighted avg       1.00      1.00      1.00       275
```

3. Water Quality Dataset

3. A

In this section of the problem we cleaned up the missing values and replaced with the mean value of that column instead of putting it zero at all the places because mean might be able to give us more accurate decision.

In [133…

```
waterquality = pd.read_csv("water_potability.csv")
waterquality.head()
```

Out[133…

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalom
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31

In [134...
```
waterquality = waterquality.replace('NaN', np.nan)
waterquality = waterquality.fillna(waterquality.mean()
waterquality.head()
```

Out[134...

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalom
0	7.080795	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86
1	3.716080	129.422921	18630.057858	6.635246	333.775777	592.885359	15.180013	56
2	8.099124	224.236259	19909.541732	9.275884	333.775777	418.606213	16.868637	66
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31

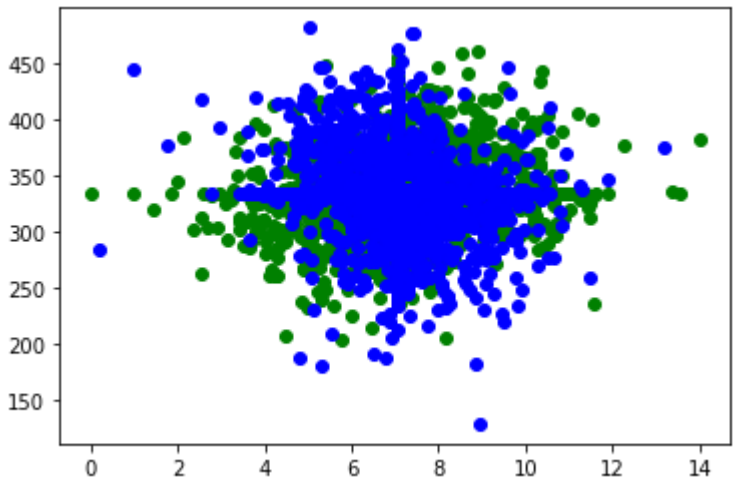
In [135...
```
waterquality0 = waterquality[waterquality.Potability==0]
waterquality1 = waterquality[waterquality.Potability==1]
waterquality1.head()
```

Out[135...

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalc
250	9.445130	145.805402	13168.529156	9.444471	310.583374	592.659021	8.606397	
251	9.024845	128.096691	19859.676476	8.016423	300.150377	451.143481	14.770863	
252	7.080795	169.974849	23403.637304	8.519730	333.775777	475.573562	12.924107	
253	6.800119	242.008082	39143.403329	9.501695	187.170714	376.456593	11.432466	
254	7.174135	203.408935	20401.102461	7.681806	287.085679	315.549900	14.533510	

In [147...
```
plt.scatter(waterquality0['ph'], waterquality0['Sulfate'], color='green')
plt.scatter(waterquality1['ph'], waterquality1['Sulfate'], color='blue')
```

Out[147...
```
<matplotlib.collections.PathCollection at 0x225cc11af70>
```



3. B, C & D

In this section we were suppose to find the linear model and also non linear model of the SVM. Plus with the accurcy report and the confusion matrix for both of them. But when we run SVM linear it run but did not produce confusion

matrix and accuracy report and below is a example of non linear model of SVM.

In [155…
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split

array = waterquality.values
X = array[:, 0:9]
y = array[:,9]

X_train_water,X_test_water,y_train_water,y_test_water = train_test_split(X, y, test_siz
X_train_water.shape
```

Out[155… (2620, 9)

In [166…
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt2

X_train_water,X_test_water,y_train_water,y_test_water = train_test_split(X, y, test_siz

m1 = LinearRegression()

# Train the model with training data
m1.fit(X_train_water, y_train_water)

# Make predictions on test data
linear_y_pred3 = m1.predict(X_test_water)

linear_y_pred4 = m1.predict(X_train_water)
```

In [168…
```
X_train_water,X_test_water,y_train_water,y_test_water = train_test_split(X, y, test_siz
```

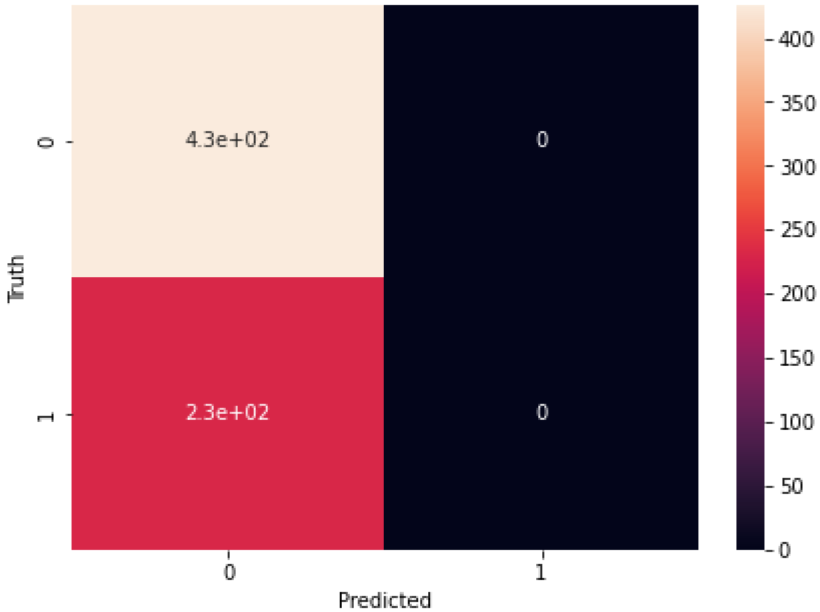
In [158…
```
modelnonlinear = SVC()
```

In [159…
```
modelnonlinear.fit(X_train_water,y_train_water)
```

Out[159… SVC()

In [140…
```
y_predict = modelnonlinear.predict(X_test_water)
```

In [141…
```
cm = confusion_matrix(y_test_water, y_predict)
plt1.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt1.xlabel('Predicted')
plt1.ylabel('Truth')
plt1.show()
```



In [144…

```
from sklearn.metrics import classification_report
print(classification_report(y_test_water,y_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.65      1.00      0.79       426
         1.0       0.00      0.00      0.00       230

    accuracy                           0.65       656
   macro avg       0.32      0.50      0.39       656
weighted avg       0.42      0.65      0.51       656

C:\Users\neelp\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Unde
finedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\neelp\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Unde
finedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\neelp\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Unde
finedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

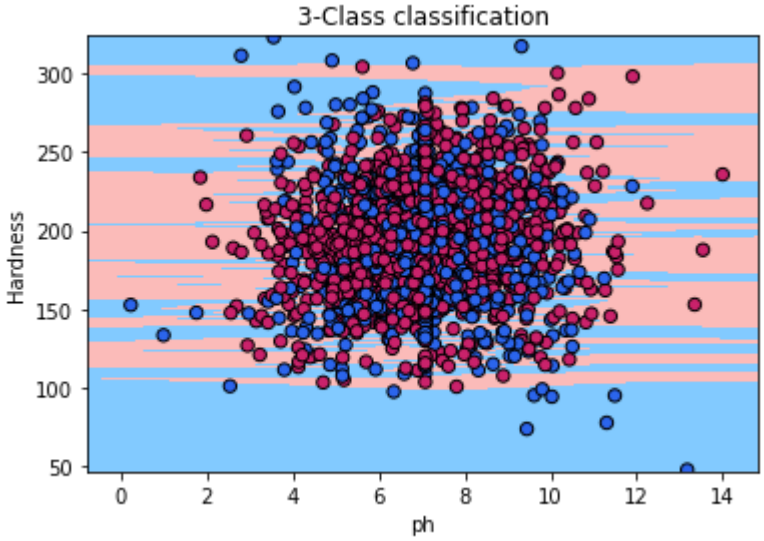
3. E

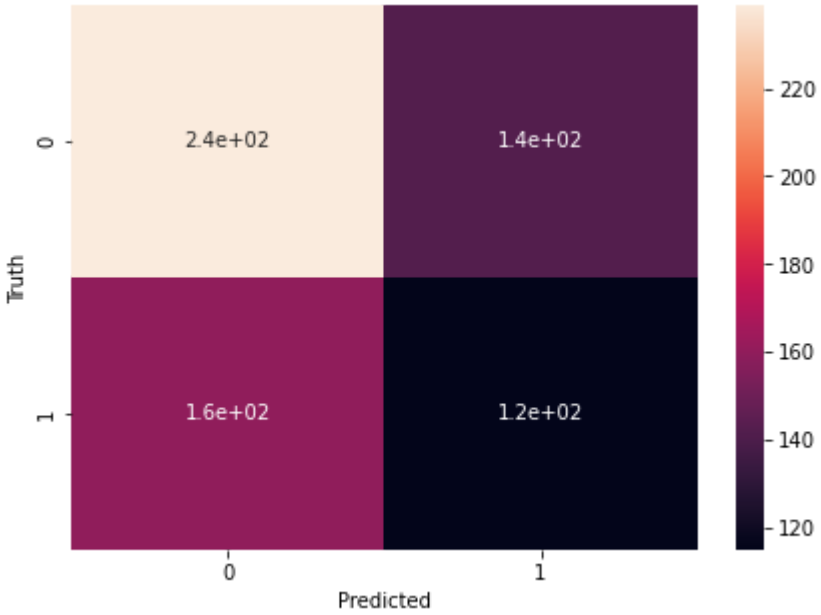
Use you own knn model and scikitlearn knn model on the water quality dataset. Report both model performance and compare them.

In [161…

```
X_train_water,X_test_water,y_train_water,y_test_water = train_test_split(X, y, test_siz
xtrain = X_train_water[:, :2]
xtest = X_test_water[:, :2]

cmap_light = ListedColormap(['#FBBBB9', '#5EFB6E', '#82CAFF'])
cmap_bold = ListedColormap(['#CA226B', '#387C44', '#2B65EC'])
cmap_test = ListedColormap(['#8E35EF', '#FFFF00', '#659EC7'])

#meshstep size parameter
```

```
h = 0.2

#KNN learner
model = KNeighborsClassifier(1)


#Fitting the data
model.fit(xtrain, y_train_water)
y_pred = model.predict(xtest)

# Plot the decision boundary
# For using meshgrid, you need to find the min max values of both attributes
# We usually make min/max a little lower/higher than the actual value
# here y is representing the second attributes, do not confuse it with the label
x_min, x_max = xtrain[:, 0].min() - 1, xtrain[:, 0].max() + 1
y_min, y_max = xtrain[:, 1].min() - 1, xtrain[:, 1].max() + 1


# make the meshgrid
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),np.arange(y_min, y_max, h))
# add the classifier to the meshgrid
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)


# plot the outcome
plt.figure()
plt.pcolormesh(xx, yy, Z, cmap=cmap_light, shading='auto')
plt.scatter(xtrain[:, 0], xtrain[:, 1], c=y_train_water, cmap=cmap_bold,edgecolor='k',
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xlabel("ph")
plt.ylabel("Hardness")
plt.title("3-Class classification")
plt.show()

cm = confusion_matrix(y_test_water, y_pred)
plt1.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt1.xlabel('Predicted')
plt1.ylabel('Truth')
plt1.show()

from sklearn.metrics import classification_report
print(classification_report(y_test_water,y_pred))
```





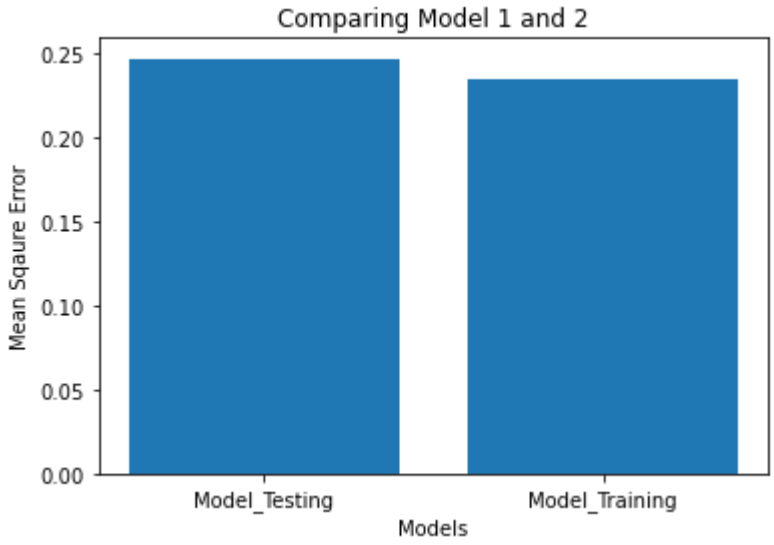
	precision	recall	f1-score	support
0.0	0.60	0.63	0.61	381
1.0	0.45	0.42	0.43	275
accuracy			0.54	656
macro avg	0.52	0.52	0.52	656
weighted avg	0.54	0.54	0.54	656

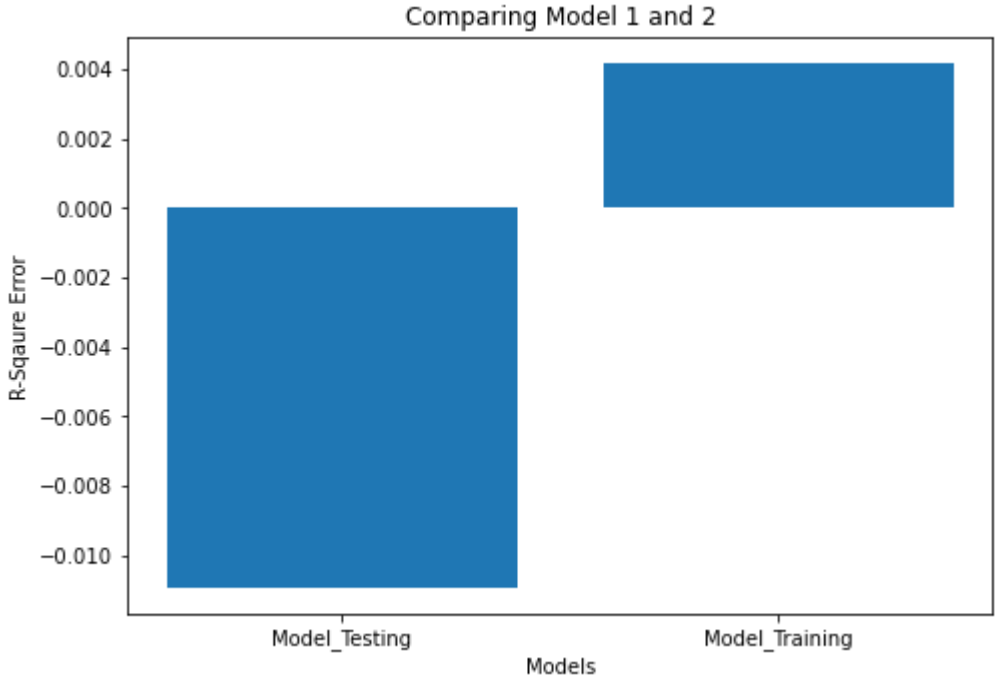
In [165…

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt2

X_train_water,X_test_water,y_train_water,y_test_water = train_test_split(X, y, test_siz

m1 = LinearRegression()

# Train the model with training data
m1.fit(X_train_water, y_train_water)
```

```
# Make predictions on test data
diabetes_y_pred3 = m1.predict(X_test_water)

diabetes_y_pred4 = m1.predict(X_train_water)

#print the mean squared error
print('training mean squared error: %.2f'% mean_squared_error(y_train_water, diabetes_y

# # print the r-squared
print('training R-squared: %.2f' % r2_score(y_train_water, diabetes_y_pred4))

#print the mean squared error
print('testing mean squared error: %.2f'% mean_squared_error(y_test_water, diabetes_y_p


# print the r-squared
print('testing R-squared: %.2f' % r2_score(y_test_water, diabetes_y_pred3))

models = ['Model_Testing','Model_Training']
MSElist = [mean_squared_error(y_test_water, diabetes_y_pred3), mean_squared_error(y_tra
Rlist = [r2_score(y_test_water, diabetes_y_pred3), r2_score(y_train_water, diabetes_y_p

# Plot
fig = plt.figure()
plt.bar(models,MSElist)
plt.xlabel("Models")
plt.ylabel("Mean Sqaure Error")
plt.title("Comparing Model 1 and 2")
plt.show()

fig = plt2.figure()
ax = fig.add_axes([0,0,1,1])
plt2.bar(models, Rlist)
plt2.xlabel("Models")
plt2.ylabel("R-Sqaure Error")
plt2.title("Comparing Model 1 and 2")
plt2.show()
```

```
training mean squared error: 0.24
training R-squared: 0.00
testing mean squared error: 0.25
testing R-squared: -0.01
```





END OF Project 3