DATA7703 Assignment 1

2022 Semester 2

Question 1

(a) In this question, I took "BODY MASS" as the independent variable and "SVL" as the dependent variable, which I also indicate them as the horizontal and vertical coordinate labels in question (d).

Equation: $SVL = 0.05873624 * BODY\ MASS + 64.0456745$
SSE: 29081.74531785096

(b) Quadratic regression is an extension of simple linear regression. While linear regression can be performed with as few as two points, quadratic regression come with the disadvantage that it requires more data points to be certain your data falls into the "U" shape which will be more visually illustrated in the picture in question (d).

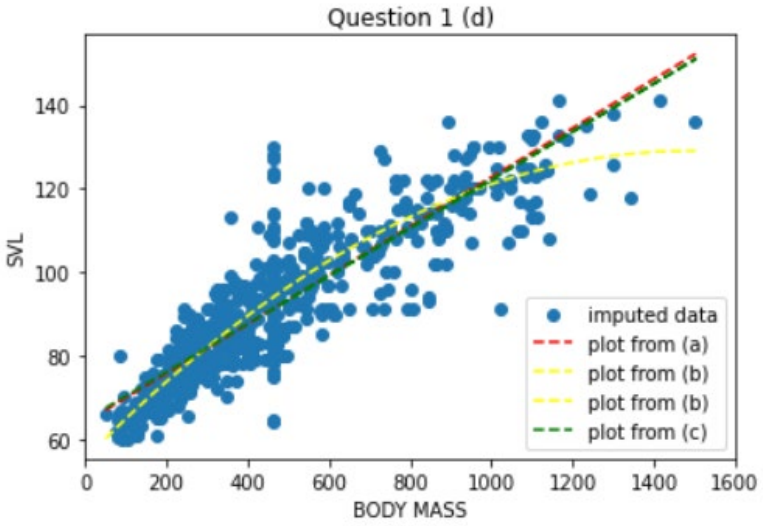
Equation: $SVL = -3.32117126e^{-5} * BODY\ MASS^2 + 9.89928002e^{-2} * BODY\ MASS + 55.314286366833336$
SSE: 23805.339913386662

We can conclude that the error of this model is significantly less than the first one.

(c) After substituting the missing values with mean values, the equation and SSE are as follows:
Equation: $SVL = 0.05756542 * BODY\ MASS + 64.58082298$
SSE: 45598.581262117805

We spotted that the SSE of this model is larger than that of the model before, possibly because the sample distribution was changed by mean imputation, leading to great differences in results.

(d) The imputed data and models from (a), (b), (c) plot is shown as follows:



Question 2

Python Code:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

df2 = pd.read_csv('reg2d.csv', header=None).values
X2 = df2[:,0:2]
y2 = df2[:,2]

quadratic_featurizer2 = PolynomialFeatures(degree=2, include_bias=False)
X2_quadratic = quadratic_featurizer2.fit_transform(X2)

reg_quadratic2 = LinearRegression().fit(X2_quadratic, y2)
print("Linear model's coefficient values of Question 2: ",
reg_quadratic2.coef_,"Intercept:" , reg_quadratic2.intercept_)
```

Results:

```
Linear model's coefficient values of Question 2:  [-0.25697386  0.05128251
1.14226452  0.13806308  0.8996328 ] Intercept: 0.060413633787995136
```

Question 3

(a) In this sub-question, I am using the sample mean of the values for a given column and substitute this for the missing values.
Python code:

```
import pandas as pd
import numpy as np

df3 = pd.read_csv('penguins_size.csv')
mass_average = np.mean(df3['body_mass_g'])
length_average = np.mean(df3['flipper_length_mm'])

df3.loc[:,'body_mass_g'] = df3['body_mass_g'].fillna(mass_average)
df3.loc[:,'flipper_length_mm'] = df3['flipper_length_mm'].fillna(length_average)
df3.head()
```

Result:

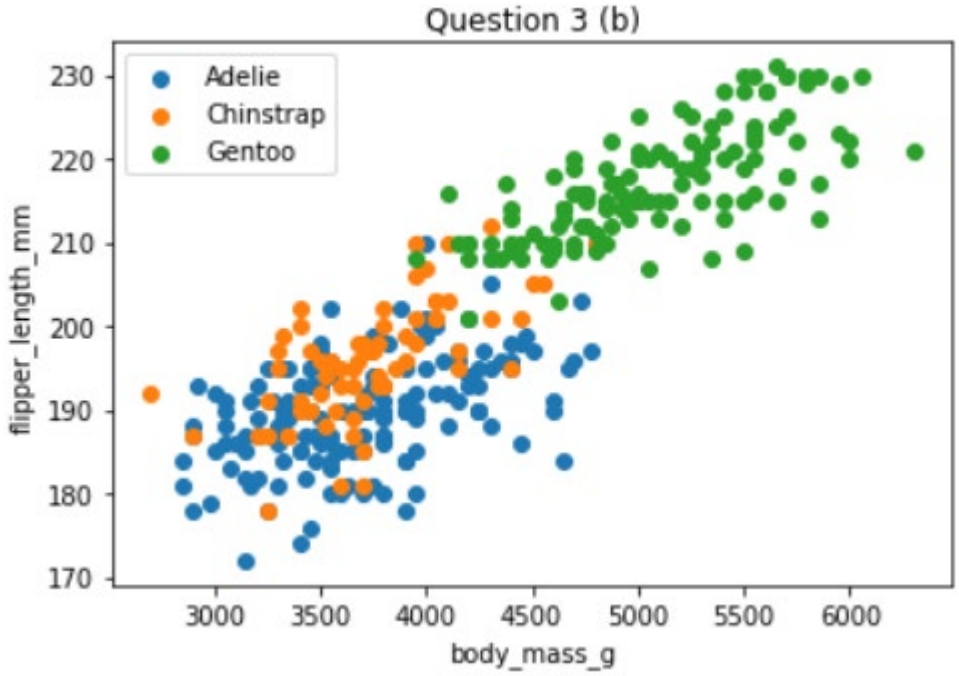
	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.000000	3750.000000	MALE
1	Adelie	Torgersen	39.5	17.4	186.000000	3800.000000	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.000000	3250.000000	FEMALE
3	Adelie	Torgersen	NaN	NaN	200.915205	4201.754386	NaN
4	Adelie	Torgersen	36.7	19.3	193.000000	3450.000000	FEMALE

We can clearly see that in row “3”, "culmen_length_mm", "culmen_depth_mm" and "sex" are still “Nah”, while the NaH values for "flipper_length_mm" and "body_mass_g" have been completed.

(b) Python Code:

```
import matplotlib.pyplot as plt

plt.scatter(df3[df3['species']=='Adelie']['body_mass_g'],
df3[df3['species']=='Adelie']['flipper_length_mm'], label='Adelie')
plt.scatter(df3[df3['species']=='Chinstrap']['body_mass_g'],
df3[df3['species']=='Chinstrap']['flipper_length_mm'], label='Chinstrap')
plt.scatter(df3[df3['species']=='Gentoo']['body_mass_g'],
df3[df3['species']=='Gentoo']['flipper_length_mm'], label='Gentoo')

plt.title("Question 3 (b)")
plt.xlabel('body_mass_g')
plt.ylabel('flipper_length_mm')
plt.legend()
```

Result:



(c) Python Code:

```
from sklearn.model_selection import train_test_split
X3 = df3.iloc[:,4:6]
y3 = df3.iloc[:,0]

X_train, X_test, y_train, y_test = train_test_split(X3, y3, test_size=0.30,
random_state=617)

print(X3.shape, X_train.shape, X_test.shape)
print(y3.shape, y_train.shape, y_test.shape)
```

Result:

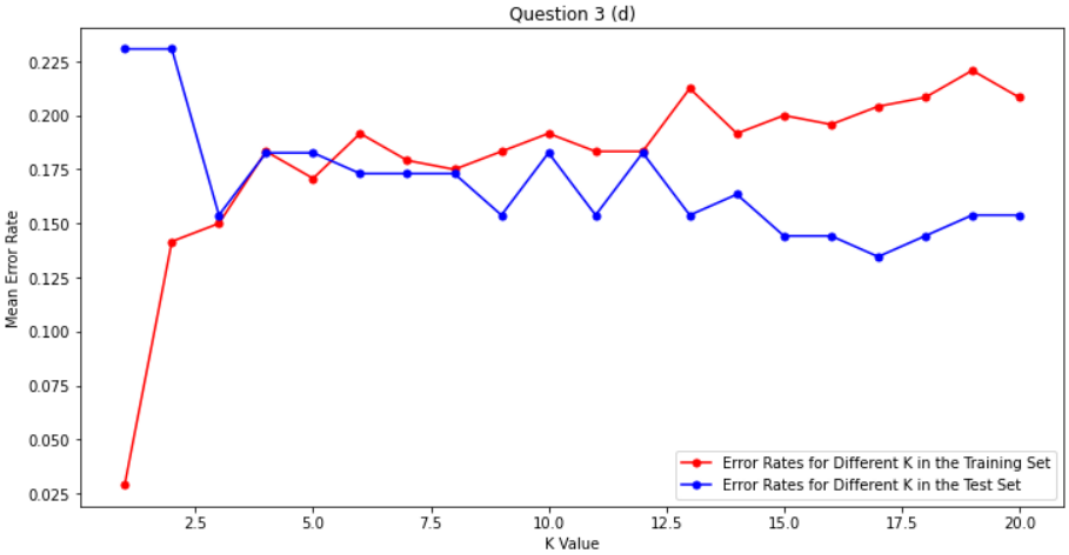
```
(344, 2) (240, 2) (104, 2)
(344,) (240,) (104,)
```

Since we have 344 rows of data and 344*0.7≈240.8, we should have 240 rows of data for X_train & y_train, while 344-240 = 104 rows of data for X_test & y_test, as shown in the output.

(d) Python Code:

```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

scaler = StandardScaler()
scaler.fit(X_train)

# Perform standardization by centering and scaling
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

train_error = []
test_error = []

# Calculating error for K values between 1 and 21
for i in range(1, 21):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)

    pred_i_train = knn.predict(X_train)
    train_error.append(np.mean(pred_i_train != y_train))
    pred_i_test = knn.predict(X_test)
    test_error.append(np.mean(pred_i_test != y_test))

plt.figure(figsize=(12, 6))
plt.plot(range(1, 21), train_error, color='red', marker='o', markersize=5, label='Error
Rates for Different K in the Training Set')
plt.plot(range(1, 21), test_error, color='blue', marker='o', markersize=5, label='Error
Rates for Different K in the Test Set')

plt.title('Question 3 (d)')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
plt.legend()
```

Result:



(e) Python Code:

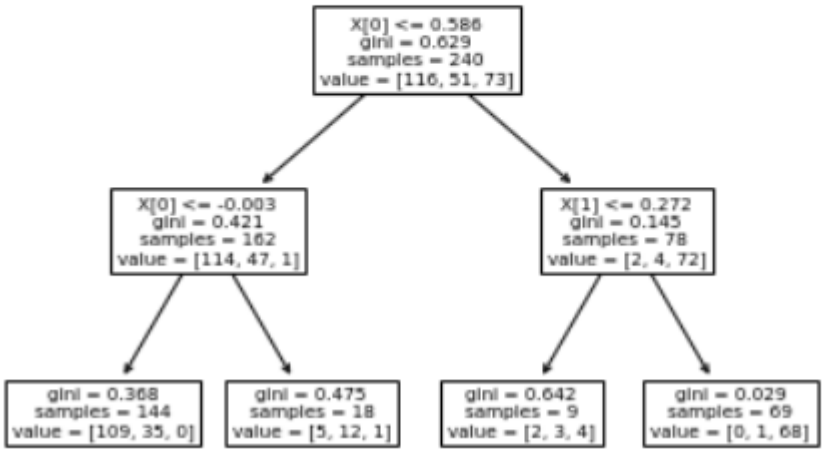
```
from sklearn import tree
from sklearn.metrics import accuracy_score
tree_model = tree.DecisionTreeClassifier(criterion='gini')
tree_model.fit(X_train, y_train)

Ypred_tree_train = tree_model.predict(X_train)
Ypred_tree_test = tree_model.predict(X_test)

tree.plot_tree(tree_model)
print("The training set errors rate is: " , 1 - accuracy_score(Ypred_tree_train, y_train))
print("The test set errors rate is: " , 1 - accuracy_score(Ypred_tree_test, y_test))
```

Result:

```
The training set errors rate is:  0.029166666666666674
The test set errors rate is:  0.22115384615384615
```

It seems that this decision tree is over-fitting, we cannot even see the criteria for each node clearly, so I modify the code:
"tree_model = tree.DecisionTreeClassifier(criterion='gini', max_depth=2, min_samples_leaf=2)"
in order to make the tree more readable. The result is as follows:

```
The training set errors rate is:  0.1958333333333333
The test set errors rate is:  0.17307692307692313
```



But in fact, this may lead to the under-fitting result. For example, the leftmost Leaf node has 144 samples in total, but obviously it can be subdivided further. Unfortunately, I am not able to solve this problem at present.