Supplementary Code

Subject:

Machine Learning

Section:

A

Submitted to:

Dr. Bilal Wajid

Submitted by:

Hamza Arif (F2019313011)

Saira Tariq (F2019313054)

M.Fahed Chaudhry (F2019313046)

Farhat-ul-ain (F2019313048)

Contents

1- NBB4 Plasmid

Importing Libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import mean_squared_error
from math import sqrt
```

Importing Datasets

```
dataset = pd.read_excel (r'C:\Users\92305\Desktop\Project
ML\Extra\NBB4\NBB4.xlsx')
```

Creating sets

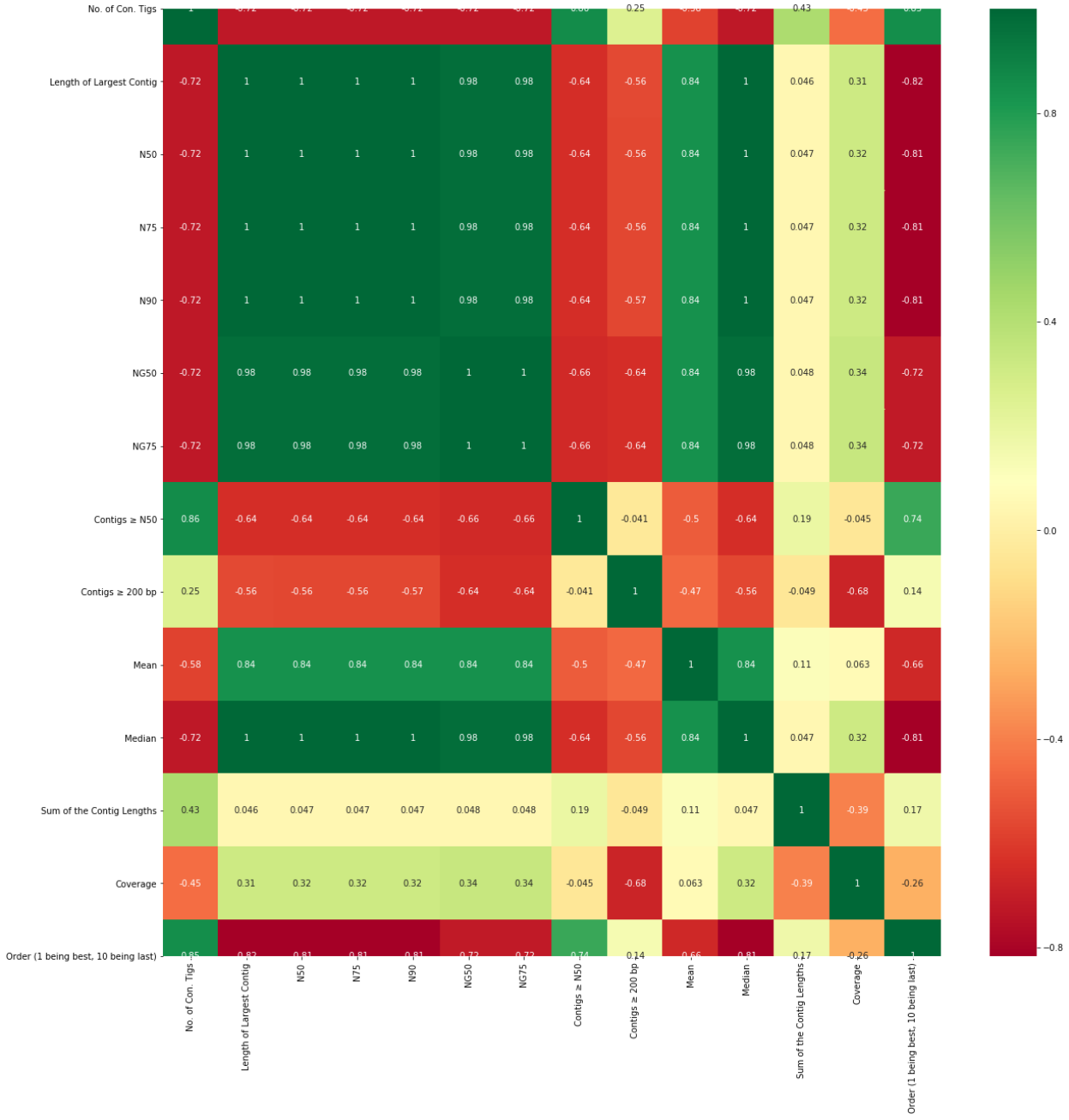
```
datasets  = dataset.iloc[:,0:1]
X0 = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 14].values
```

Encoding Varaible

```
from sklearn.preprocessing import LabelEncoder
labelencode_X = LabelEncoder()
X0[:,0] = labelencode_X.fit_transform(X0[:,0])
```

FeatuerSelection

```
from sklearn.feature_selection import SelectKBest
import seaborn as sns
#get correlations of each features in dataset
corrmat = dataset.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(dataset[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



Adding Selected Featuers to X

```
X0 = dataset[['No. of Con. Tigs','Contigs ≥ N50']]
```

SplitDataintoTrain and Test

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X0,y, test_size = 0.2,
random_state = 0)
```

Fitting Multiple Linear Regression to training set

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

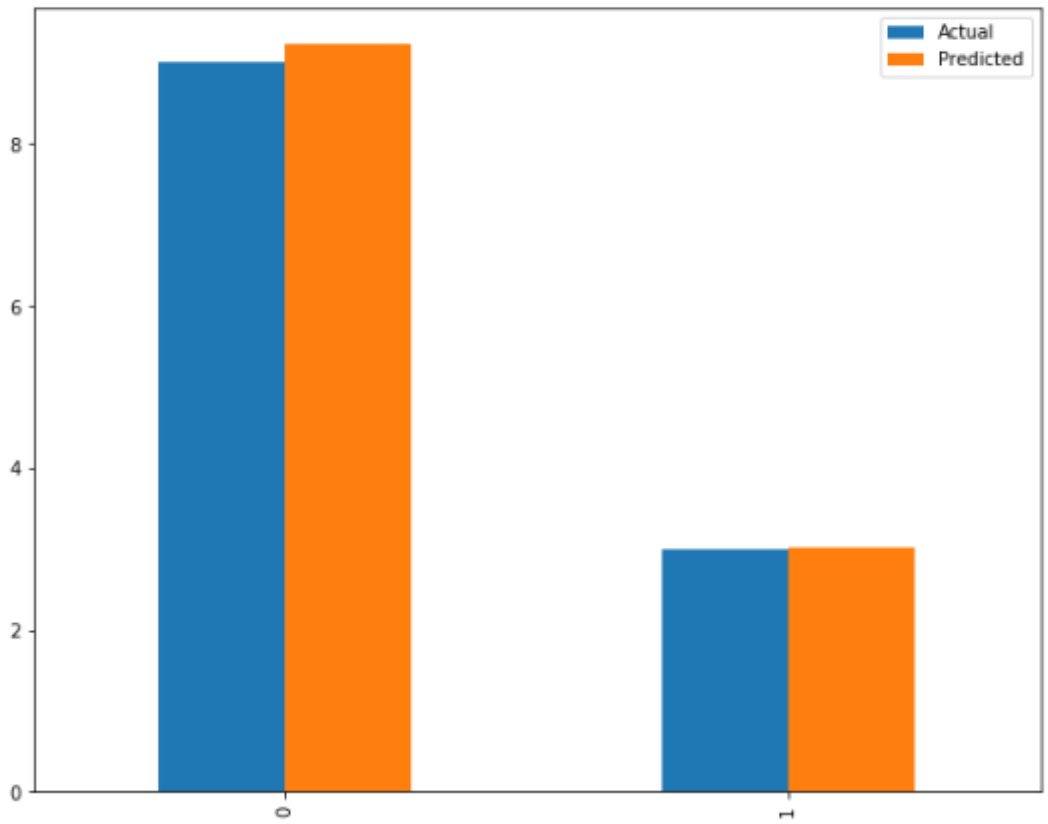
Prediciting test set result

```
y_pred = regressor.predict(X_test)
regressor.score(X_test, y_test)

#0.9971770719168614
```

Actual vs Predicted

```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df.plot(kind='bar',figsize=(10,8))
df
```

	Actual	Predicted
0	9	9.225310
1	3	3.006927



Predicting Order with out LinearModel

```
y_hat= regressor.predict(X0)
order_linear = pd.DataFrame(y_hat, columns=['Predicted Order'])
order_linear['Assembly Metrics'] = datasets
order_linear.sort_values(by= 'Predicted Order' )
```

	Predicted Order	Assembly Metrics
7	2.638222	Mira2
9	2.638222	MARAGAP
8	3.006927	Maq
5	3.898435	IDBA
0	4.045268	Velvet
6	6.021571	Mira
4	6.785851	SHARCGS
3	7.873882	QSRA
1	9.098550	VCAKE
2	9.225310	SSAKE

2- Hamburgensis X14

Importing Datasets

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing DataSet

```
dataset1 = pd.read_excel (r'C:\Users\92305\Desktop\Project
ML\Extra\Hamburgenis\HamburgenisiTranspose.xlsx')
```

Creating sets

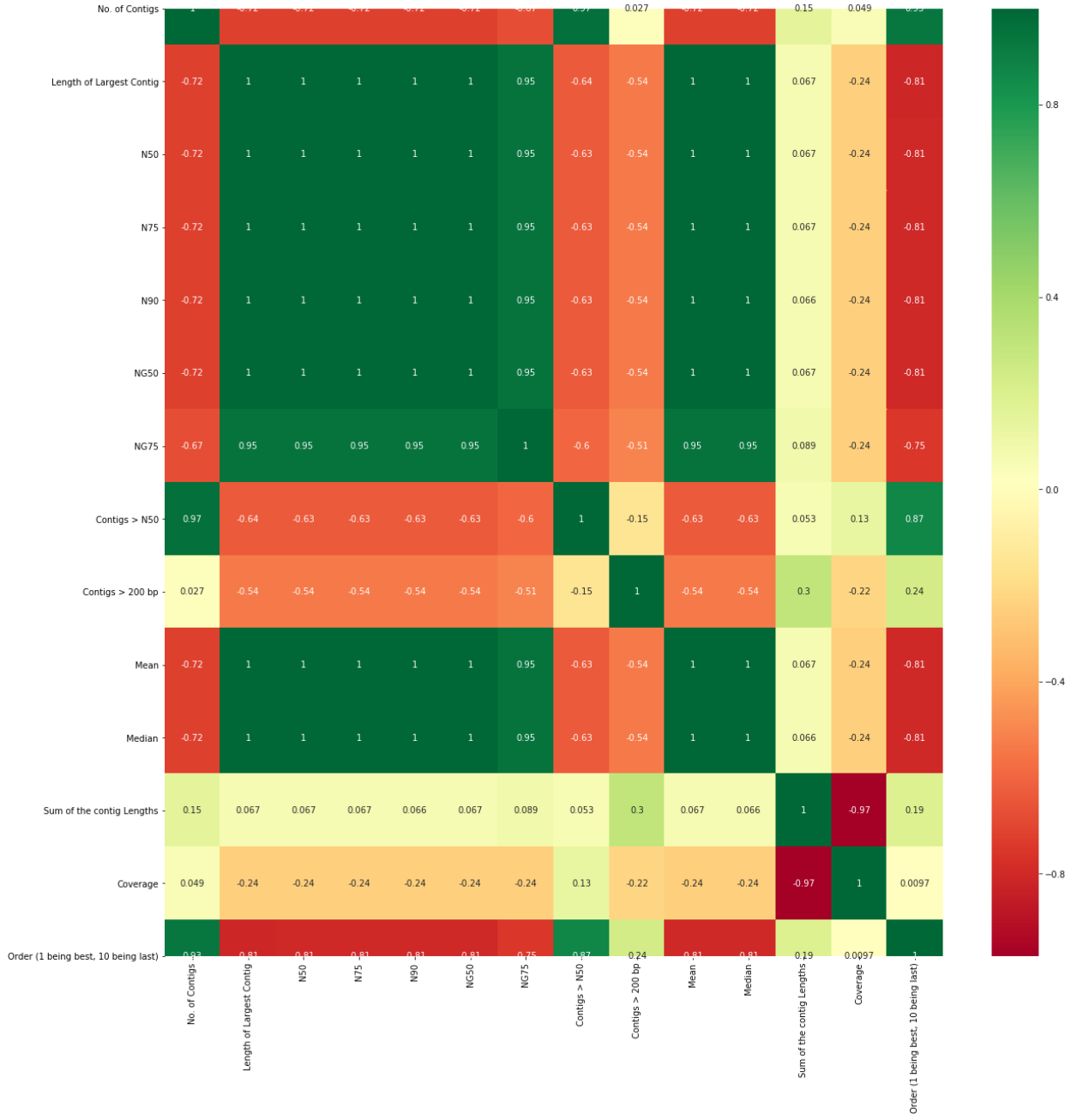
```
datasets1  = dataset1.iloc[:,0:1]
X1 =(dataset1.iloc[:, :-1].values)
y1 = (dataset1.iloc[:, 14].values)
```

Encoding Varaible

```
from sklearn.preprocessing import LabelEncoder

labelencode_X1 = LabelEncoder()
X1[:,0] = labelencode_X1.fit_transform(X1[:,0])
```

FeatuerSelection

```
from sklearn.feature_selection import SelectKBest
import seaborn as sns
#get correlations of each features in dataset
corrmat1 = dataset1.corr()
top_corr_features = corrmat1.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(dataset1[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



Adding Selected Featuers to X

```
X1 = dataset1[['No. of Contigs','Contigs > N50']].values
```

SplitDataintoTrain and Test

```
from sklearn.model_selection import train_test_split
X1_train, X1_test, y1_train, y1_test = train_test_split(X1,y1, test_size =
0.2, random_state = 0)
```

Fitting Multiple Linear Regression to training set

```
from sklearn.linear_model import LinearRegression
regressor1 = LinearRegression()
regressor1.fit(X1_train, y1_train)
```

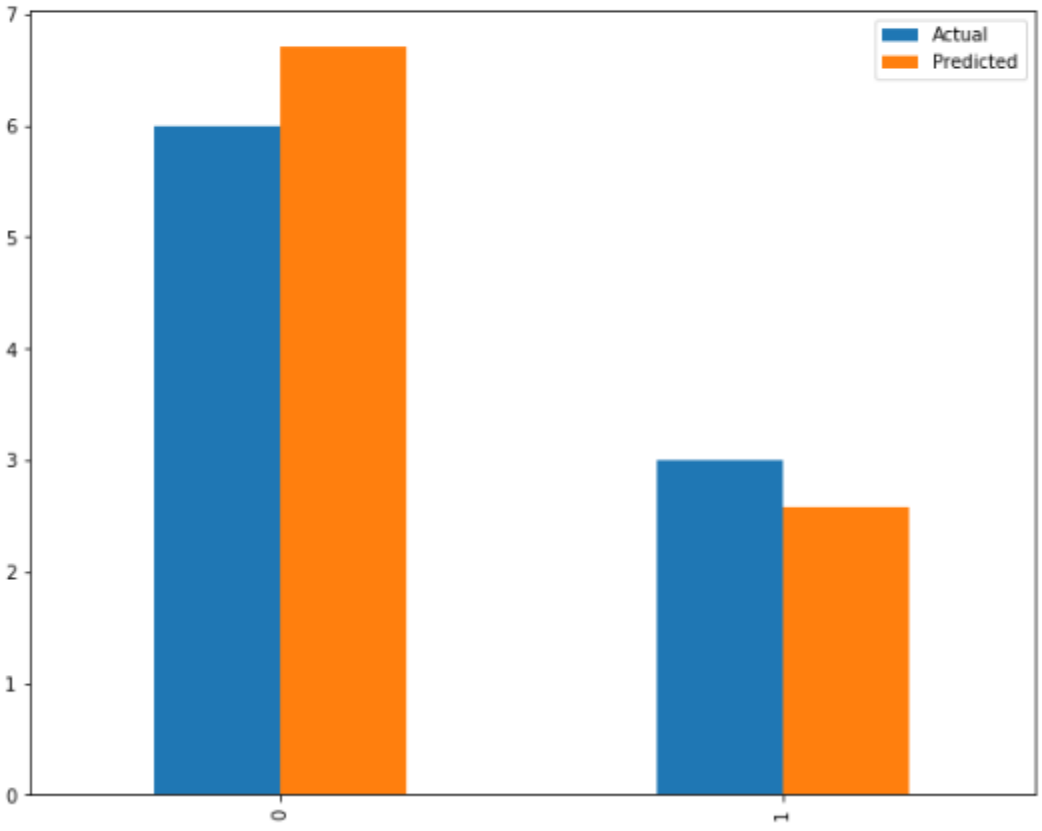
Prediciting test set result

```
y1_pred = regressor1.predict(X1_test)
regressor1.score(X1_test, y1_test)

#0.8517637797884923
```

Actual vs Predicted

```
df1 = pd.DataFrame({'Actual': y1_test, 'Predicted': y1_pred})
df1.plot(kind='bar',figsize=(10,8))
df1
```

	Actual	Predicted
0	6	6.702820
1	3	2.583938



Predicting Order with out LinearModel

```
y_hat1= regressor1.predict(X1)
order_linear = pd.DataFrame(y_hat1, columns=['Predicted Order'])
order_linear['Assembly Metrics'] = datasets1
order_linear.sort_values(by= 'Predicted Order' )
```

	Predicted Order	Assembly Metrics
7	2.299910	Mira2
9	2.299910	MARAGAP
8	2.583938	Maq
0	3.234427	Velvet
5	4.046115	IDBA
2	6.702820	SSAKE
4	7.638747	SHARCGS
1	7.908924	VCAKE
3	8.924955	QSRA
6	9.647012	Mira

3- Vibrio Cholerae

Importing Datasets

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing Dataset

```
dataset2 = pd.read_excel (r'C:\Users\92305\Desktop\Project ML\Vibrio Cholerae
Transpose.xlsx')
```

Creating sets

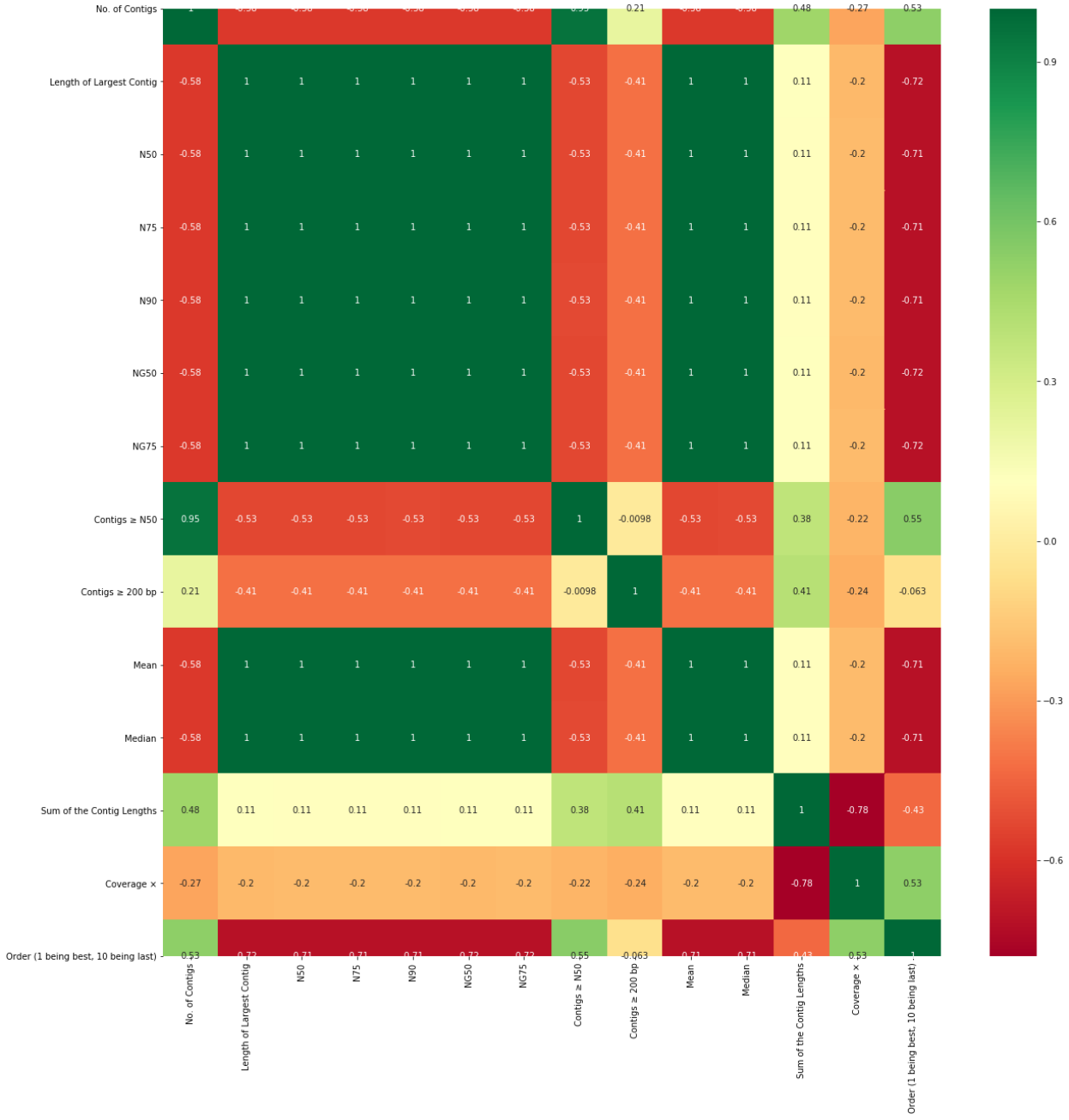
```
datasets2  = dataset2.iloc[:,0:1]
X2 = dataset2.iloc[:, :-1].values
y2 = dataset2.iloc[:, 14].values
```

Encoding Varaible

```
from sklearn.preprocessing import LabelEncoder , OneHotEncoder,
OrdinalEncoder

labelencode_X2 = LabelEncoder()
X2[:,0] = labelencode_X2.fit_transform(X2[:,0])
```

FeatuerSelection

```
from sklearn.feature_selection import SelectKBest
import seaborn as sns
#get correlations of each features in dataset
corrmat2 = dataset2.corr()
top_corr_features = corrmat2.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(dataset2[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



Adding Selected Featuers to X

```
X2 = dataset2[['No. of Contigs','Contigs ≥ N50','Contigs ≥ 200 bp','Sum of the Contig Lengths']]
```

SplitDataintoTrain and Test

```
from sklearn.model_selection import train_test_split
X2_train, X2_test, y2_train, y2_test = train_test_split(X2,y2, test_size = 0.2, random_state = 0)
```

Fitting Multiple Linear Regression to training set

```
from sklearn.linear_model import LinearRegression
regressor2 = LinearRegression()
regressor2.fit(X2_train, y2_train)
```

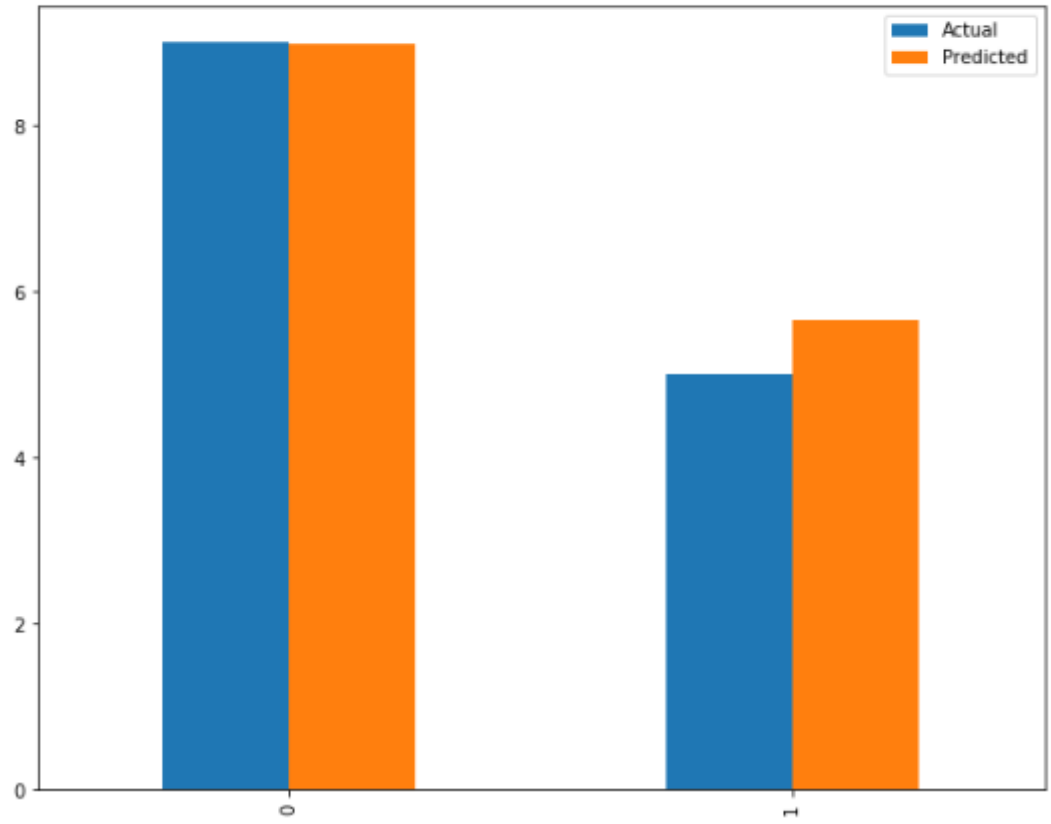
Prediciting test set result

```
y2_pred = regressor2.predict(X2_test)
regressor2.score(X2_test, y2_test)
```

Actual vs Predicted

```
df2 = pd.DataFrame({'Actual': y2_test, 'Predicted': y2_pred})
df2.plot(kind='bar',figsize=(10,8))
df2
```

	Actual	Predicted
0	9	8.975946
1	5	5.656689



Predicting Order with out LinearModel

```
y_hat2= regressor2.predict(X2)
order_linear = pd.DataFrame(y_hat2, columns=['Predicted Order'])
order_linear['Assembly Metrics'] = datasets2
order_linear.sort_values(by= 'Predicted Order' )
```

	Predicted Order	Assembly Metrics
9	1.562475	MARAGAP
7	1.606290	Mira2
0	2.639481	Velvet
5	4.706391	IDBA
8	5.656689	Maq
1	6.148571	VCAKE
6	7.008486	Mira
3	7.831533	QSRA
2	8.975946	SSAKE
4	9.496772	SHARCGS

4- PAb1

Importing Libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

importing Data set

```
dataset3 = pd.read_excel (r'C:\Users\92305\Desktop\Project ML\PAb1
Transpose.xlsx')
```

Creating sets

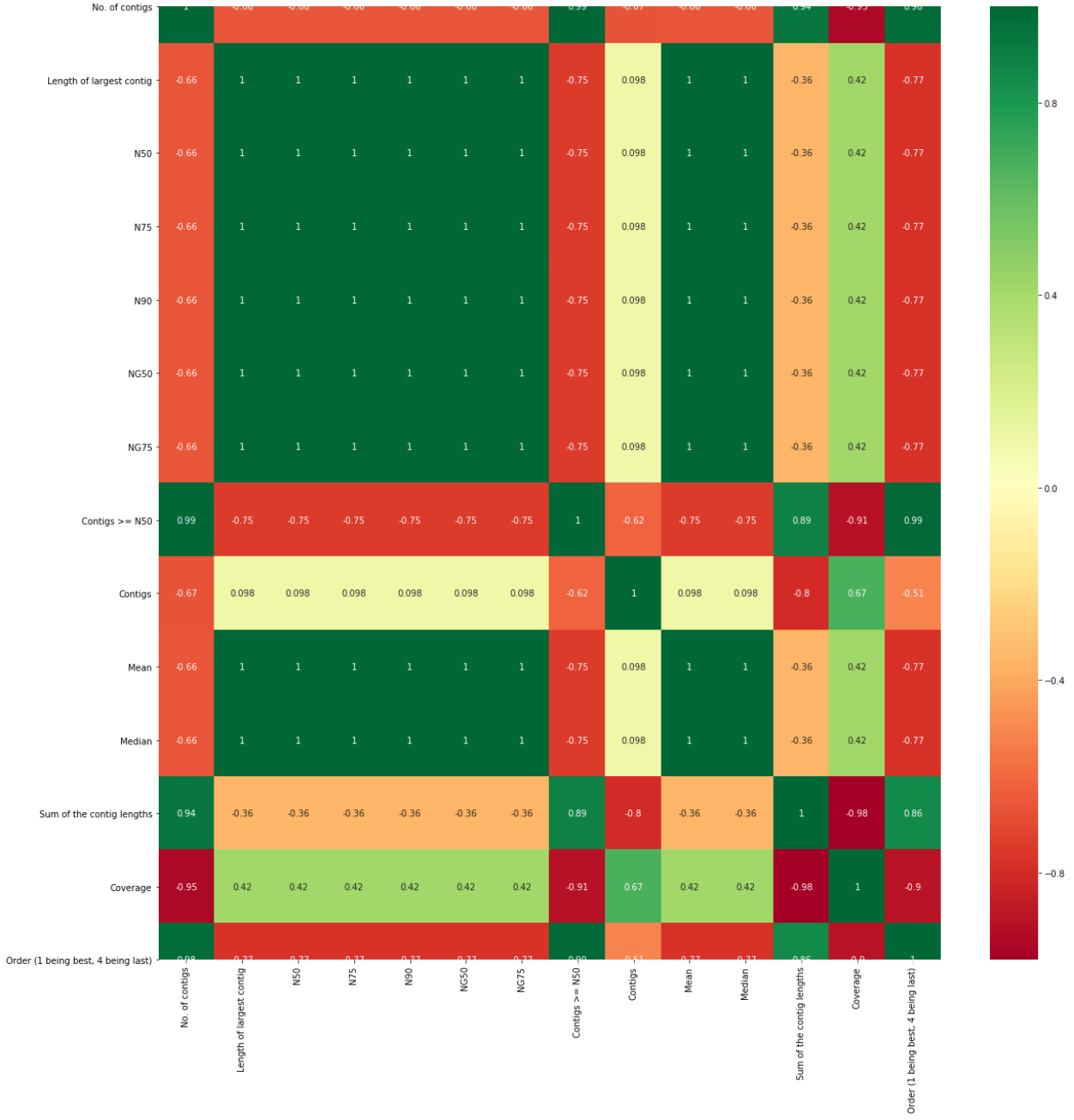
```
datasets3= dataset3.iloc[:,0:1]
X3 = dataset3.iloc[:, :-1].values
y3 = dataset3.iloc[:, 14].values
```

Encoding Varaible

```
from sklearn.preprocessing import LabelEncoder

labelencode3_X = LabelEncoder()
X3[:,0] = labelencode3_X.fit_transform(X3[:,0])
```

FeatuerSelection

```
from sklearn.feature_selection import SelectKBest
import seaborn as sns
#get correlations of each features in dataset
corrmat3 = dataset3.corr()
top_corr_features = corrmat3.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(dataset3[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



Adding Selected Featuers to X

```
X3 = dataset3[['No. of contigs',' Contigs >= N50','Sum of the contig lengths']]
```

SplitDataintoTrain and Test

```
from sklearn.model_selection import train_test_split
X3_train, X3_test, y3_train, y3_test = train_test_split(X3,y3, test_size = 0.2, random_state = 0)
```

Fitting Multiple Linear Regression to training set

```
from sklearn.linear_model import LinearRegression
regressor3 = LinearRegression()
regressor3.fit(X3_train, y3_train)
```

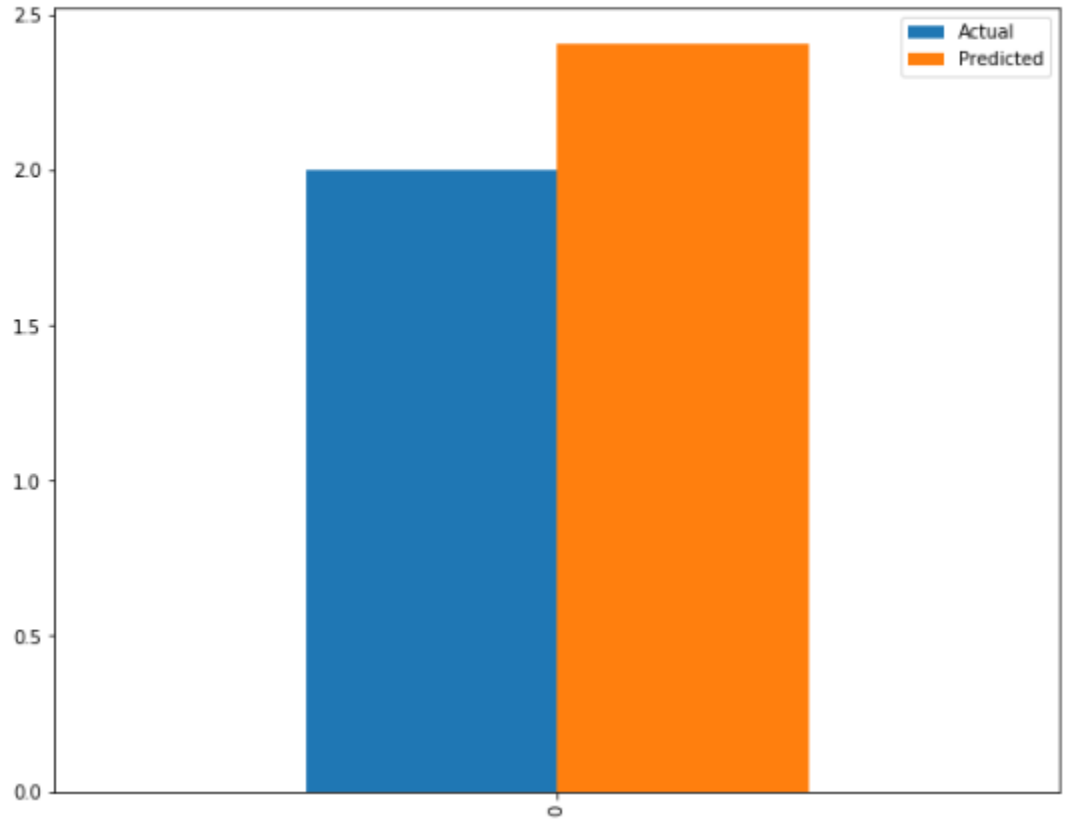
Prediciting test set result

```
y3_pred = regressor3.predict(X3_test)
regressor3.score(X3_test, y3_test)
regressor3.predict(X3_test)
```

Actual vs Predicted

```
df3 = pd.DataFrame({'Actual': y3_test, 'Predicted': y3_pred})
df3.plot(kind='bar',figsize=(10,8))
df3
```

	Actual	Predicted
0	2	2.403087



Predicting Order with out LinearModel

```
y_hat3= regressor3.predict(X3)

order_linear = pd.DataFrame(y_hat3, columns=['Predicted Order'])
order_linear['Assembly Metrics'] = datasets3
order_linear.sort_values(by= 'Predicted Order' )
```

	Predicted Order	Assembly Metrics
3	1.000000	MARAGAP
2	2.403087	IDBA
1	3.000000	QSRA
0	4.000000	VCAKE

6- Accuracy

MSE & RMSE

```
#NBB4
mse = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)
#Hamburgensis X14
mse1 = mean_squared_error(y1_test, y1_pred)
rmse1 = sqrt(mse1)
#Vibro Chlorea
mse2 = mean_squared_error(y2_test, y2_pred)
rmse2 = sqrt(mse2)
#PAb1
mse3 = mean_squared_error(y3_test, y3_pred)
rmse3 = sqrt(mse3)

print(rmse)
print(rmse1)
print(rmse2)
print(rmse3)
```

```
0.15939370360289246
0.5775218571412621
0.4646607717368355
0.40308712126706325
```

reg.score(X, y)

```
print(regressor.score(X_test, y_test))
print(regressor1.score(X1_test, y1_test))
print(regressor2.score(X2_test, y2_test))
# The result for PAb1 will be "NaN" (R^2 is not calculated because dataset is
small)
print(regressor3.score(X3_test, y3_test))
```

```
0.9971770719168614
0.8517637797884923
0.9460225918022321
nan
```

7- Predicted Order

NBB4 Plasmid

```
order_linear = pd.DataFrame(y_hat, columns=['Predicted Order'])
order_linear['Assembly Metrics'] = datasets
order_linear.sort_values(by= 'Predicted Order' )
```

	Predicted Order	Assembly Metrics
7	2.638222	Mira2
9	2.638222	MARAGAP
8	3.006927	Maq
5	3.898435	IDBA
0	4.045268	Velvet
6	6.021571	Mira
4	6.785851	SHARCGS
3	7.873882	QSRA
1	9.098550	VCAKE
2	9.225310	SSAKE

Hamburgenisis X14

```
order_linear = pd.DataFrame(y_hat1, columns=['Predicted Order'])
order_linear['Assembly Metrics'] = datasets1
order_linear.sort_values(by= 'Predicted Order' )
```

	Predicted Order	Assembly Metrics
7	2.299910	Mira2
9	2.299910	MARAGAP
8	2.583938	Maq
0	3.234427	Velvet
5	4.046115	IDBA
2	6.702820	SSAKE
4	7.638747	SHARCGS
1	7.908924	VCAKE
3	8.924955	QSRA
6	9.647012	Mira

Vibro Chlorea

```
order_linear = pd.DataFrame(y_hat2, columns=['Predicted Order'])
order_linear['Assembly Metrics'] = datasets2
order_linear.sort_values(by= 'Predicted Order' )
```

	Predicted Order	Assembly Metrics
9	1.562475	MARAGAP
7	1.606290	Mira2
0	2.639481	Velvet
5	4.706391	IDBA
8	5.656689	Maq
1	6.148571	VCAKE
6	7.008486	Mira
3	7.831533	QSRA
2	8.975946	SSAKE
4	9.496772	SHARCGS

PAb1

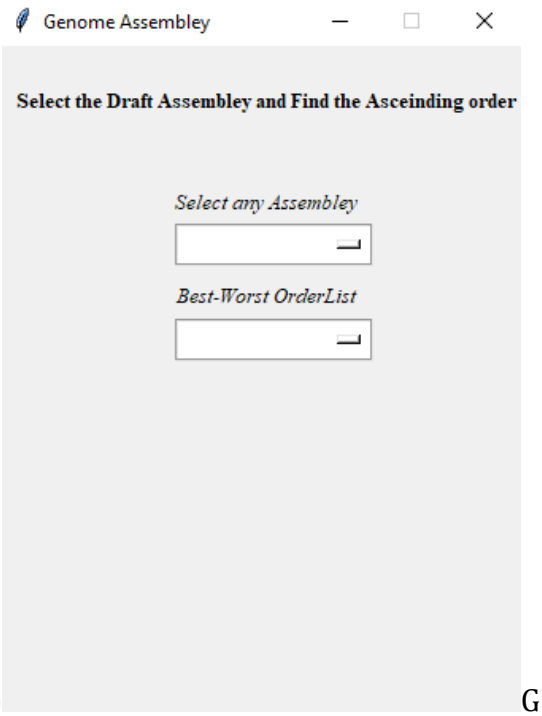
```
order_linear = pd.DataFrame(y_hat3, columns=['Predicted Order'])
order_linear['Assembly Metrics'] = datasets3
order_linear.sort_values(by= 'Predicted Order' )
```

	Predicted Order	Assembly Metrics
3	1.000000	MARAGAP
2	2.403087	IDBA
1	3.000000	QSRA
0	4.000000	VCAKE

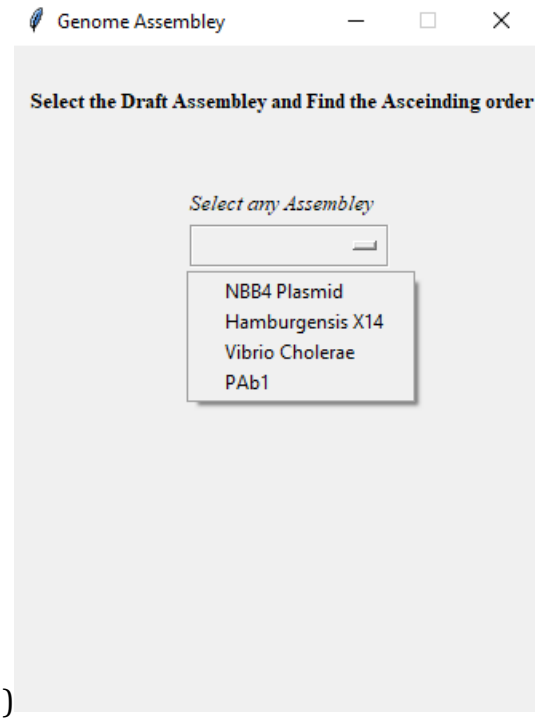
8- GUI

```
from tkinter import *
from PIL import ImageTk, Image
from tkinter import ttk

Canv = Tk()
Canv.title("Show List of Draft Assemblies")
Canv.geometry("330x420+50+50")

ttk.Label(Canv, text = "Select the Draft Assembley and Find the Asceinding
order",  font = ("Times New Roman", 10,'bold')).grid(column = 0,  row = 15,
padx = 10, pady = 25)
on_BU_change = ttk.Combobox(Canv, width = 27, exportselection=False)
def on_BU_change(BU_selected):
    # remove current options in sector combobox
    menu = sector_drop['menu']
    menu.delete(0, 'end')
    # create new options for sector combobox based on selected value of BU
combobox
    if BU_selected == 'NBB4 Plasmid':
        selected_sectors = ["1-Mira2","2-MARAGAP","3-Maq","4-IDBA","5-
Velvet","6-SHARCGS","7-QSRA","8-VCAKE","9-SSAKE","10-Mira"]
    elif BU_selected == 'Hamburgensis X14':
        selected_sectors = ['1-Mira2','2-MARAGAP','3-Maq','4-Velvet','5-
IDBA','6-SSAKE','7-QSRA','8-VCAKE','9-SHARCGS','10-Mira']
    elif BU_selected == 'Vibrio Cholerae':
        selected_sectors = ['1-MARAGAP','2-Mira2','3-Velvet','4-IDBA','5-
Maq','6-QSRA','7-VCAKE','8-Mira','9-SSAKE','10-SHARCGS']
```

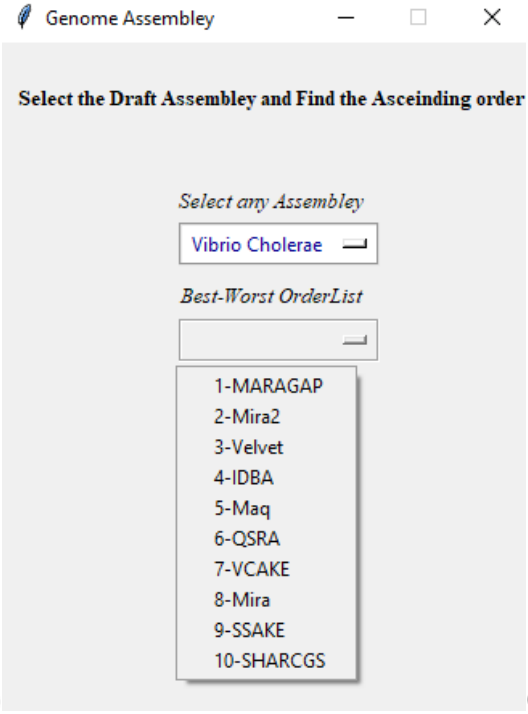
```
    elif BU_selected == 'PAb1':
        selected_sectors = ['1-MARAGAP','2-IDBA','3-QSRA','4-VCAKE']
    else:
        selected_sectors = ['']
    # clear the current selection of sector combobox
    sector.set('')
    # setup the sector combobox
    for item in selected_sectors:
        menu.add_command(label=item, command=lambda x=item:
on_sector_change(x))

BU = StringVar()
ttk.Label(Canv, text = "Select any Assembley",  font = ("Times New Roman",
10,'italic')).grid(column = 0,  row = 20, padx = 14, pady = 20)
BU_choices = ['NBB4 Plasmid', 'Hamburgensis X14', 'Vibrio Cholerae', 'PAb1']
BU_drop = OptionMenu(Canv, BU, *BU_choices, command=on_BU_change)
BU_drop_label = Label(Canv, bg="ivory", fg="darkgreen")
BU_drop.config(bg='white', fg='dark blue', width=14, relief=GROOVE,
cursor='hand2')
BU_drop.place(x=110, y=110)

def on_sector_change(sector_selected):
    sector.set(sector_selected[0])

sector = StringVar()
#sector_label = Label(Canv, text="Answer").grid(row=0,column=1)
ttk.Label(Canv, text = "Best-Worst OrderList",  font = ("Times New Roman",
10, 'italic')).grid(column = 0,  row = 25, padx = 14, pady = 20)
sector_drop = OptionMenu(Canv, sector, '', command=on_sector_change )
sector_drop.config(bg='white', fg='dark blue', width=14, relief=GROOVE,
cursor='hand2')
sector_drop.place(x=110, y=170)

Canv.title("Genome Assembley")
Canv.resizable(0,0)
Canv.mainloop()
```



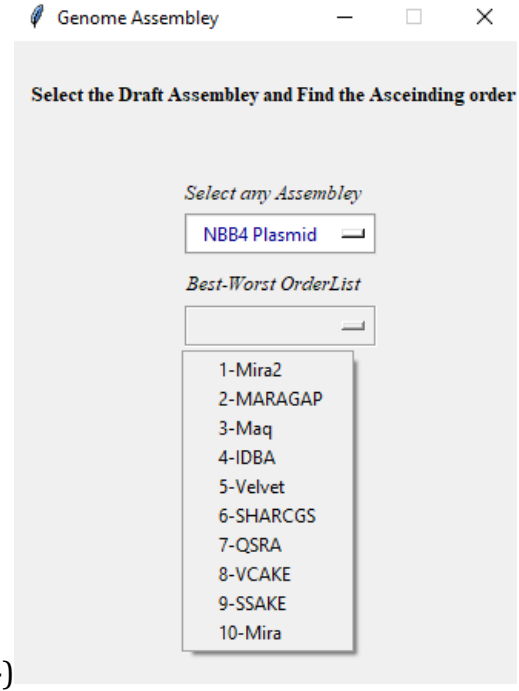
GUI(1)



GUI(2)



GUI(3)



GUI(4)