MODELS_REPORT

1.0.1 Importing Required Packages

In [16]:
```python
# NumPy is a powerful numerical computing library in Python
import numpy as np
# Pandas is a powerful data manipulation and analysis library for Python
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
```

In [17]:
```python
# Read the Codon_usage_clean_data dataset into a pandas DataFrame object
# Setting low_memory=False because it disables the memory optimization and m
df1 = pd.read_csv('Group_17_Clean_Dataset.csv', low_memory=False)
```

In [18]:
```python
# Displaying the first few rows of a DataFrame
df1.head()
```

Out[18]:

	Unnamed: 0	UUC	UUA	UUG	CUU	CUC	CUA	CUG	AUU
0	0	789	0.00050	0.00351	0.01203	0.03208	0.00100	0.04010	0.00551
1	1	938	0.00068	0.00678	0.00407	0.02849	0.00204	0.04410	0.01153
2	2	1750	0.01357	0.01543	0.00782	0.01111	0.01028	0.01193	0.02283
3	3	1815	0.01619	0.00992	0.01567	0.01358	0.00940	0.01723	0.02402
4	4	952	0.00767	0.03679	0.01380	0.00548	0.00473	0.02076	0.02716

5 rows × 65 columns

1.0.2 Importing KNeighborsClassifier and LogisticRegression from sklearn

In [19]:
```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
```

```python
# For splitting the data into trainning and testing and for hyperparameter t
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, GridSearchCV,Randomized
# For confusion_matrix, classification_report
from sklearn.metrics import roc_auc_score, roc_curve, confusion_matrix, clas
```

In [20]:
```python
# selecting all rows and all columns except the last one from DataFrame df2
X = df1.iloc[:,:-1]
# selecting the 'Kingdom_cat' column from DataFrame df2 and assigns it to y
y = df1['Kingdom_cat']
# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,
```

In [21]:
```python
import warnings
warnings.filterwarnings('ignore')
```

1.0.3 KNeighborsClassifier

In [22]:
```python
from sklearn.metrics import classification_report

# Adjust classification report to avoid warnings
print(classification_report(y_test, y_pred, zero_division=1))
```

```
              precision    recall  f1-score   support

           0       1.00      0.00      0.00        25
           1       0.22      0.27      0.24       584
           2       0.17      0.07      0.10       269
           3       1.00      0.00      0.00       114
           4       1.00      0.00      0.00        44
           5       1.00      0.00      0.00         4
           6       0.22      0.23      0.23       505
           7       1.00      0.00      0.00        36
           8       1.00      0.00      0.00        43
           9       0.21      0.22      0.22       567
          10       0.17      0.26      0.20       415

    accuracy                           0.20      2606
   macro avg       0.64      0.10      0.09      2606
weighted avg       0.28      0.20      0.19      2606
```

In [23]:
```python
knn = KNeighborsClassifier(n_neighbors=6) # Fit the classifier to the traini
knn.fit(X_train, y_train)
# Predict the labels of the test data: y_pred
y_pred = knn.predict(X_test)
# Generate the confusion matrix and classification report
display(confusion_matrix(y_test, y_pred))
print(classification_report(y_test,     y_pred))
print('Accuracy score of KNN Model: ',knn.score(X_test, y_test))
```

```
array([[  1,  22,   0,   0,   1,   0,   0,   0,   0,   1,   0],
       [  0, 569,   0,   0,  10,   0,   4,   0,   0,   1,   0],
       [  0,   0, 268,   0,   0,   0,   1,   0,   0,   0,   0],
       [  0,   0,   0, 114,   0,   0,   0,   0,   0,   0,   0],
       [  0,  22,   0,   0,  21,   0,   0,   0,   0,   1,   0],
       [  0,   4,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   3,   1,   0,   0,   0, 501,   0,   0,   0,   0],
       [  0,   0,   0,   2,   0,   0,   0,  34,   0,   0,   0],
       [  0,   0,   0,   6,   0,   0,   0,   5,  32,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0, 567,   0],
       [  0,   0,   4,   3,   0,   0,   0,   0,   0,   0, 408]])
              precision    recall  f1-score   support

           0       1.00      0.04      0.08        25
           1       0.92      0.97      0.95       584
           2       0.98      1.00      0.99       269
           3       0.91      1.00      0.95       114
           4       0.66      0.48      0.55        44
           5       0.00      0.00      0.00         4
           6       0.99      0.99      0.99       505
           7       0.87      0.94      0.91        36
           8       1.00      0.74      0.85        43
           9       0.99      1.00      1.00       567
          10       1.00      0.98      0.99       415

    accuracy                           0.97      2606
   macro avg       0.85      0.74      0.75      2606
weighted avg       0.96      0.97      0.96      2606

Accuracy score of KNN Model:   0.9650805832693784
```

1.0.4 **Observation:** We have achived Accuracy score of KNN Model is 0.97. For n_neighborsof 6.

In [24]:

```python
# Evaluating the performance of a machine learning model by comparing its pr
# on a test set and calculating the number of correct and incorrect predicti
test_t_1  =  pd.DataFrame([y_test.tolist(),y_pred.tolist()])
test_t_1 = test_t_1.transpose()
test_t_1 = test_t_1.rename(columns = {0:"Y_Test", 1:"Y_Prediction"},)
test_t_1['Check'] = np.where(test_t_1['Y_Test'] == test_t_1['Y_Prediction']
, 1, 0)
print("Number of test Cases: {}".format(test_t_1.Check.count()))
print('Number Correct: {} || Number of Wrong: {}'.format(test_t_1.value_coun
test_t_1.value_counts('Check')[0]))
```

```
Number of test Cases: 2606
Number Correct: 2515 || Number of Wrong: 91
```

In [25]:

```python
# Dictionary used in data preprocessing or analysis tasks.
# where it's necessary to convert abbreviated names to their full forms for
# So we are using this Dictionary for plotting heatmap.
K_names = {'arc': 'archaea',
'bct': ' bacteria',
'phg' : 'bacteriophage',
'plm': 'plasmid',
'pln': 'plant',
```

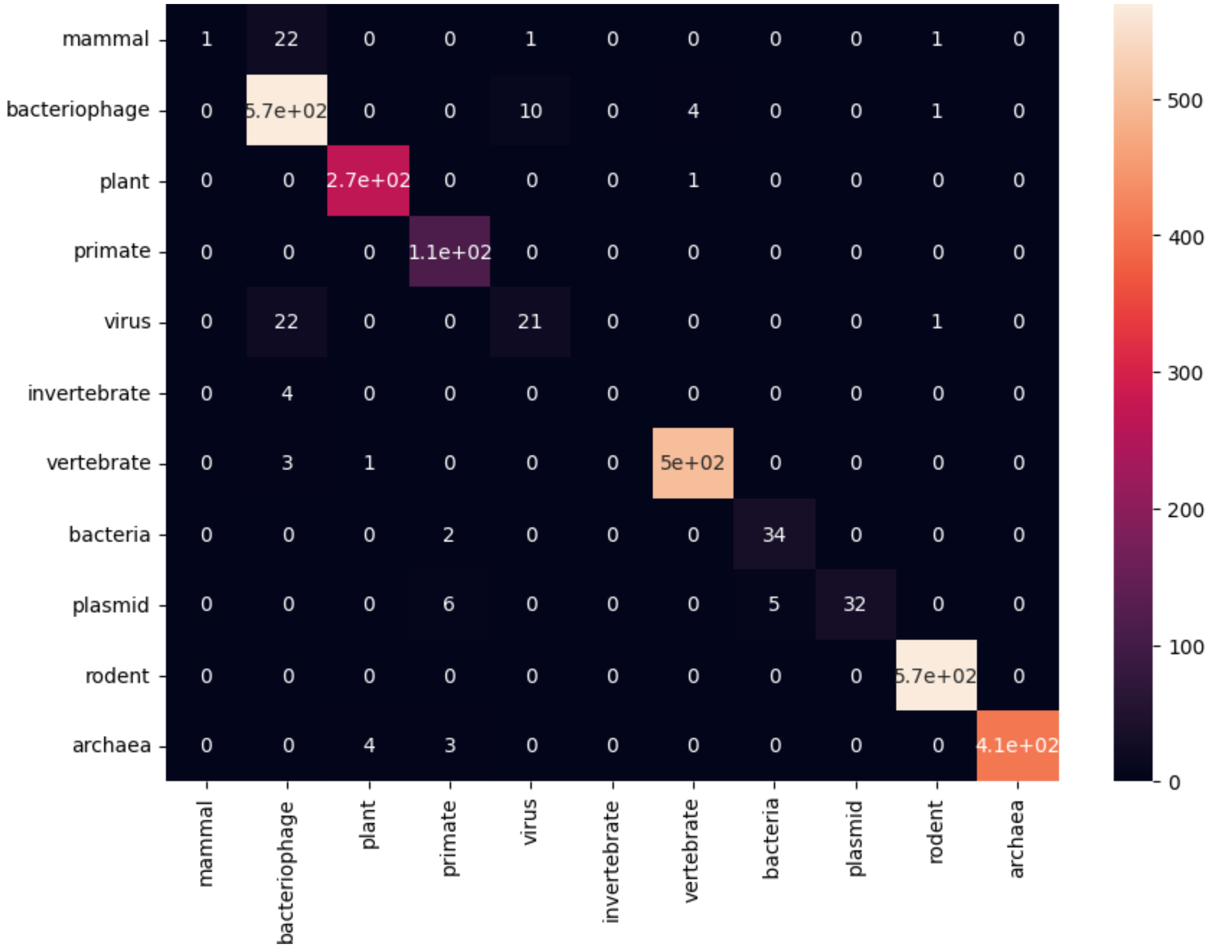
```
'inv': 'invertebrate',
'vrt': 'vertebrate',
'mam': 'mammal',
'rod': 'rodent',
'pri':  'primate',
'vrl':  'virus'}
K_names2 = {0: 'archaea',
1:  ' bacteria',
2  :  'bacteriophage',
3: 'plasmid',
4: 'plant',
5: 'invertebrate',
6: 'vertebrate',
7: 'mammal',
8: 'rodent',
9: 'primate',
10: 'virus'}
D_names  = {0:'genomic',
1:'mitochondrial',
2:'chloroplast',
3:'cyanelle',
4:'plastid',
5:'nucleomorph',
6:'secondary_endosymbiont',
7:'chromoplast', '8':'leucoplast',
9:'NA',
10:'proplastid',
11:'apicoplast',
12:'kinetoplast'}
```

1.0.5 Heatmap

In [26]:
```
# Plotting the heatmap
D_Types_a = []
for a in y_test.unique():
  D_Types_a.append(K_names2[a])
for a in y_pred:
  D_Types_a.append(K_names2[a])

D_Types_a   =   list(set(D_Types_a))


array  =  confusion_matrix(y_test,  y_pred)
df_cm = pd.DataFrame(array, index = [i for i in D_Types_a],
columns = [i for i in D_Types_a])
plt.figure(figsize  =  (10,7))
sns.heatmap(df_cm,  annot=True);
```



1.0.6 Multiclass Logistic regression

1.0.7 When we want to use Multiclass Logistic regression, we have two technique for converting Logistic regression to Multiclass Logistic regression.

1. **one-vs-one:** When there are more than two classes in the target vector, the one-vs-one strategy allows logistic regression to train a separate model for each class against every individual remaining class.

2. **one-vs-rest:** When there are more than two classes in the target vector, the one-vs-rest strategy allows logistic regression to train a separate model for each class comparing with all the remaining classes. Therefore, this is also known as the One-vs-All (OvA) strategy. In this project, we are using one-vs-rest technique. By using multi_class='ovr'

In [27]:
```
# Initializing a Logistic Regression classifier object named Lg. The multi_c
# The solver parameter is set to 'liblinear', which is suitable for classifi
Lg = LogisticRegression(multi_class='ovr', solver='liblinear')
Lg.fit(X_train, y_train)
```

Out[27]:

```
▾ LogisticRegression
LogisticRegression(multi_class='ovr', solver='liblinear')
```

In [28]:

```
from sklearn.metrics import classification_report

# Adjust classification report to avoid warnings
print(classification_report(y_test, y_pred, zero_division=1))

# Predicting the X_test
y_pred = Lg.predict(X_test)
# Generate the confusion matrix and classification report
display(confusion_matrix(y_test, y_pred))
print(classification_report(y_test,     y_pred))
print('Accuracy score of Multiclass Logistic regression:',Lg.score(X_test, y
```

```
              precision    recall  f1-score   support

           0       1.00      0.04      0.08        25
           1       0.92      0.97      0.95       584
           2       0.98      1.00      0.99       269
           3       0.91      1.00      0.95       114
           4       0.66      0.48      0.55        44
           5       1.00      0.00      0.00         4
           6       0.99      0.99      0.99       505
           7       0.87      0.94      0.91        36
           8       1.00      0.74      0.85        43
           9       0.99      1.00      1.00       567
          10       1.00      0.98      0.99       415

    accuracy                           0.97      2606
   macro avg       0.94      0.74      0.75      2606
weighted avg       0.97      0.97      0.96      2606

array([[  0,  23,   0,   0,   0,   0,   0,   0,   0,   2,   0],
       [  0, 565,   0,   0,   0,   0,   0,   0,   0,  19,   0],
       [  0,   0,  77,   0,   0,   0, 117,   0,   0,   0,  75],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 114],
       [  0,  31,   0,   0,   0,   0,   0,   0,   0,  13,   0],
       [  0,   4,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0,  87,  16,   0,   0,   0, 402,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  36],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  43],
       [  0,   4,   0,   0,   0,   0,   0,   0,   0, 563,   0],
       [  0,   0,  38,   0,   0,   0,   0,   0,   0,   0, 377]])
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	25
1	0.79	0.97	0.87	584
2	0.59	0.29	0.39	269
3	0.00	0.00	0.00	114
4	0.00	0.00	0.00	44
5	0.00	0.00	0.00	4
6	0.77	0.80	0.79	505
7	0.00	0.00	0.00	36
8	0.00	0.00	0.00	43
9	0.94	0.99	0.97	567
10	0.58	0.91	0.71	415
accuracy			0.76	2606
macro avg	0.33	0.36	0.34	2606
weighted avg	0.69	0.76	0.71	2606

Accuracy score of Multiclass Logistic regression: 0.7613200306983884

1.0.8 **Observation:** We have achived Accuracy score of KNN Model is 0.76.

1.0.9 Comparing the Multiclass Logistic regression and KNeighborsClassifier:

So, for KNeighborsClassifier we have achived accuracy score of 0.97 and Multiclass Logistic regression we have achived accuracy score of 0.76. We can clearly say that KNeighborsClassifier model is performing well for codon_usage dataset.