CSE 587
Spring 2024

DIC Project Phase 2

Group Members:

Andrew Balotin

Deep Shahane

Bhushan Mahajan

Explanation and Analysis

#Linear Regression(In class):

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. The simplest form is simple linear regression, which models the relationship between two variables (one dependent and one independent).

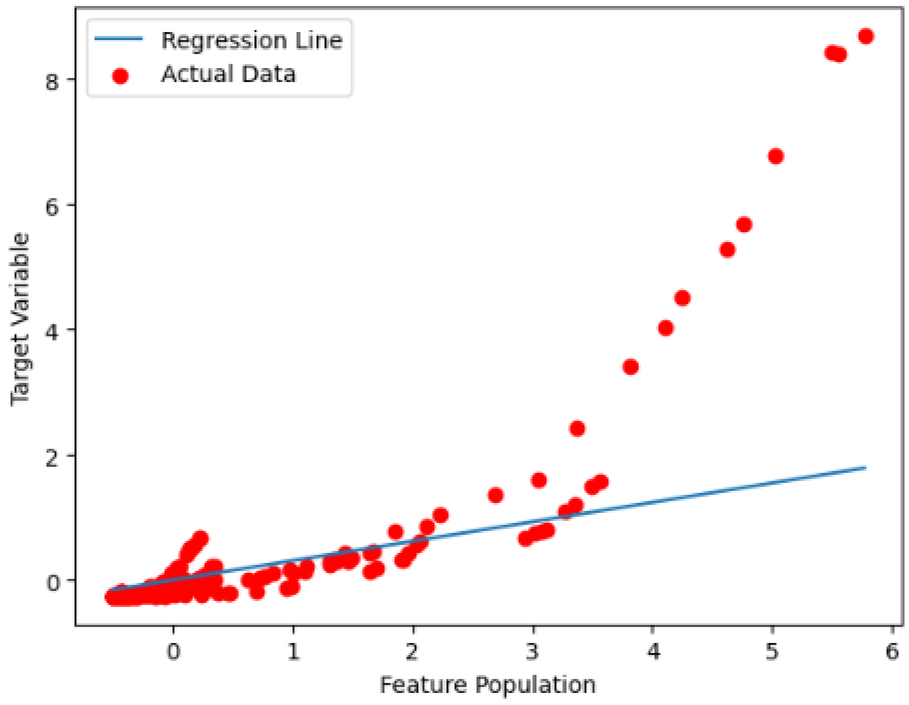
The equation for a linear regression line is typically written as $y = \boldsymbol{\beta}_0 + \boldsymbol{\beta}X_1 + \boldsymbol{\beta}X_2 \ldots . \boldsymbol{\beta}X_n$

Y is the dependent variable we're trying to predict.

$\boldsymbol{\beta}_0$ is the y-intercept of the regression line.

$\boldsymbol{\beta}X_1 \boldsymbol{\beta}X_2 \boldsymbol{\beta}X_n$ are the coefficients of the independent variables $X_1, X_2, ..X_n$ that measures the impact of each variable on y.

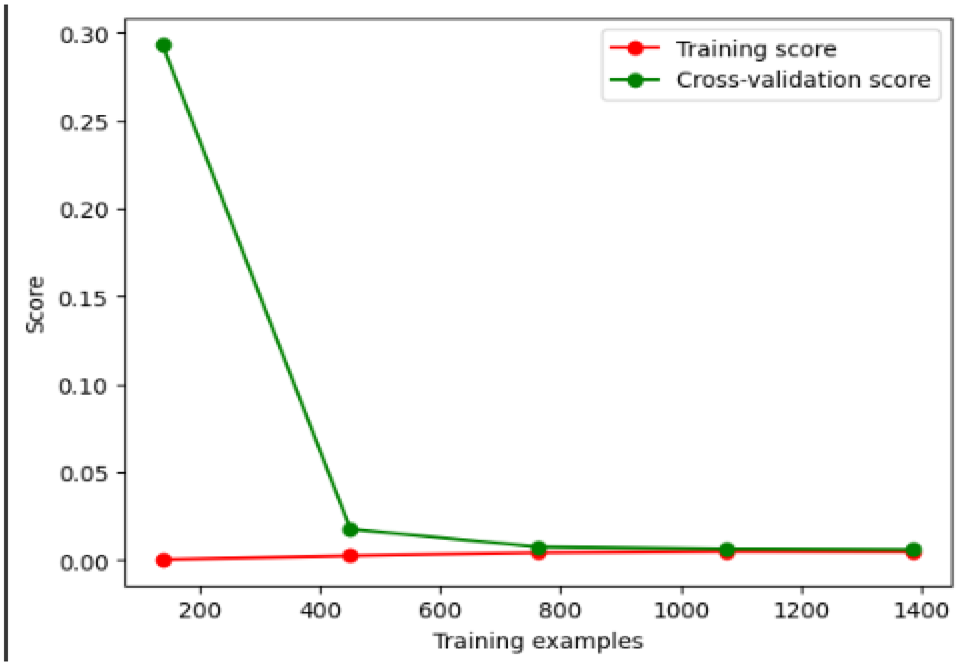
Linear regression models are straightforward to understand and interpret, making them an excellent choice for economists and policymakers who need to explain complex economic phenomena in an accessible way. The relationship between GDP and features (like coal production, nuclear production, renewable energy, energy consumption, etc) can be easily interpreted from the model coefficients and intercept. From our dataset, below is the line graph(Model regression line) we have plotted in the feature population and GDP.



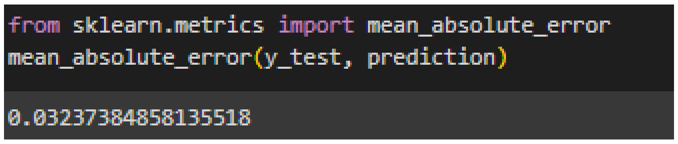
Learning Curve:

The model initially benefits from more data, but additional data doesn't improve performance after a certain point. The scores converge as the number of training examples increases. This suggests that the model may not be overfitting since both the training and validation scores are close to each other and remain stable. The plateau of both scores at a relatively high error rate suggests that adding more data may not improve the model. Instead, it could mean that the model has reached its capacity.

For scoring purposes, we have used negative mean squared error. A score that is closer to zero (less negative) indicates a better-performing model in terms of MSE.

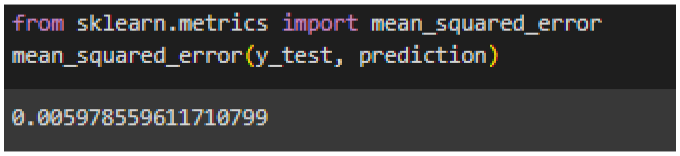


Evaluation Metrics:
Mean Absolute Error:



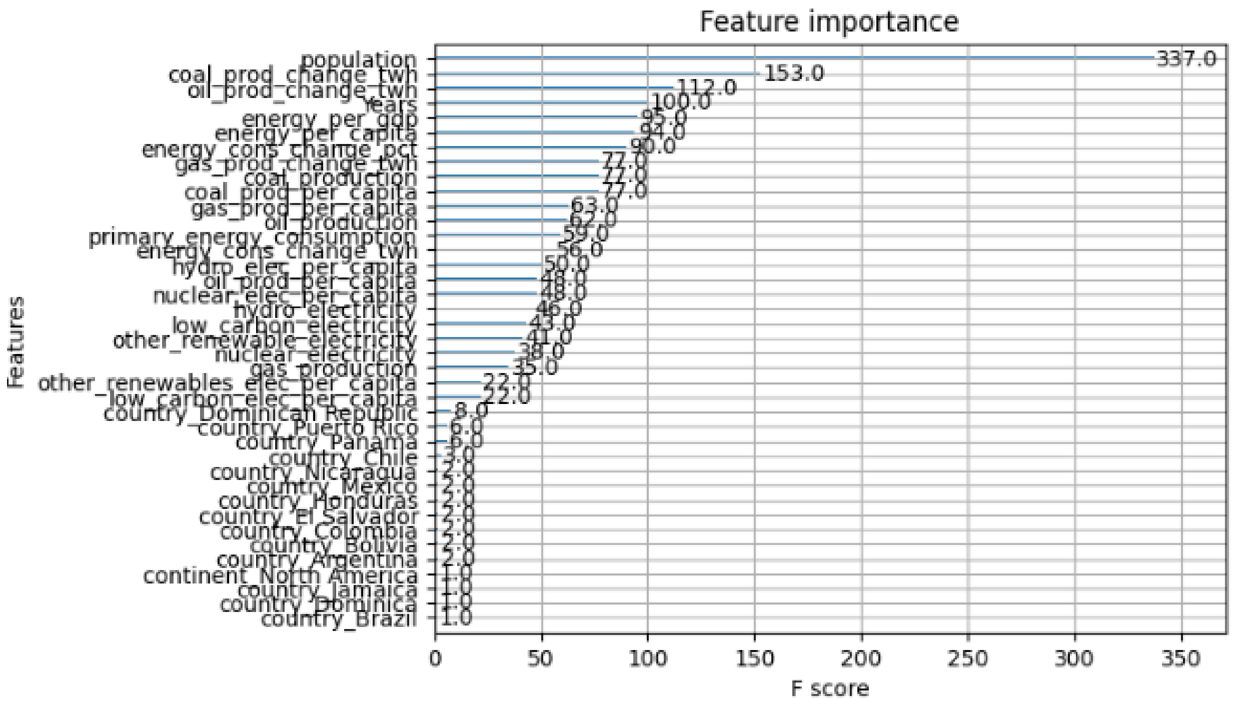
```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, prediction)

0.03237384858135518
```

Mean Squared Error:



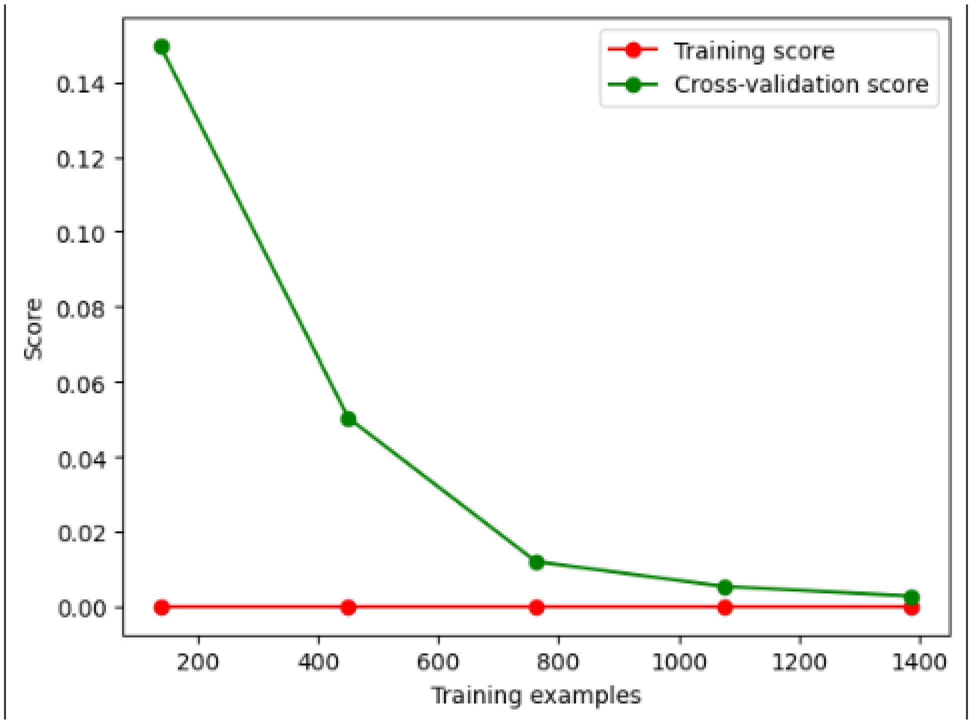
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, prediction)

0.005978559611710799
```

The error is in terms of the scaling as for training purposes we have scaled the features.
#XGboost (Outside Class):
XGBoost stands for eXtreme Gradient Boosting, which is an advanced implementation of a gradient boosting algorithm. Decision trees are commonly used as the base learners in this ensemble technique.XGBoost is designed for speed and performance. It includes L1 (Lasso Regression) and L2 (Ridge Regression) regularization to prevent overfitting, which is not available in traditional Gradient Boosting. It has an in-built feature to handle missing values.XGBoost makes splits up to the max_depth specified and then starts pruning the tree backward and removes splits beyond which there is no positive gain. For training purposes, we have set the max_depth as 5 and the L2 regularization term on weights by penalizing models with extreme coefficient values. Set the reg_lambda value of 0.4.
Feature Importance after training:

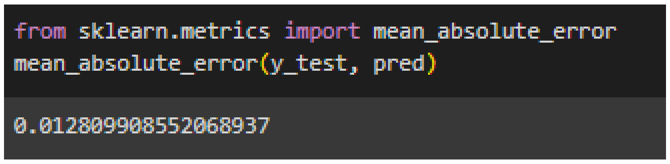


The feature importance plot will have features on the y-axis and their importance scores on the x-axis. Higher values mean the feature is more important for making the model's decisions. This visual can be very helpful in understanding which features your model relies on most to make predictions. From the graph, we can say the population feature has the highest value so first importance will be given to the feature population.

Learning Curve:



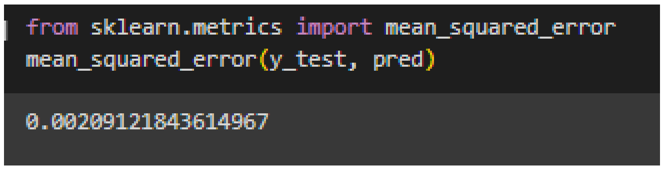
Mean Absolute Error:



```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, pred)
```

```
0.012809908552068937
```

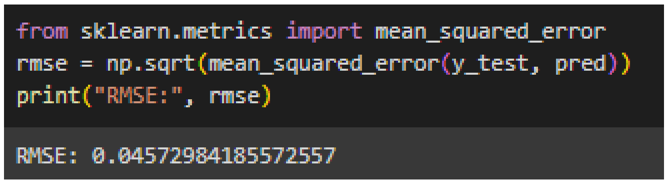
Mean Squared Error:



```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, pred)
```

```
0.00209121843614967
```

Root Mean Squared Error:

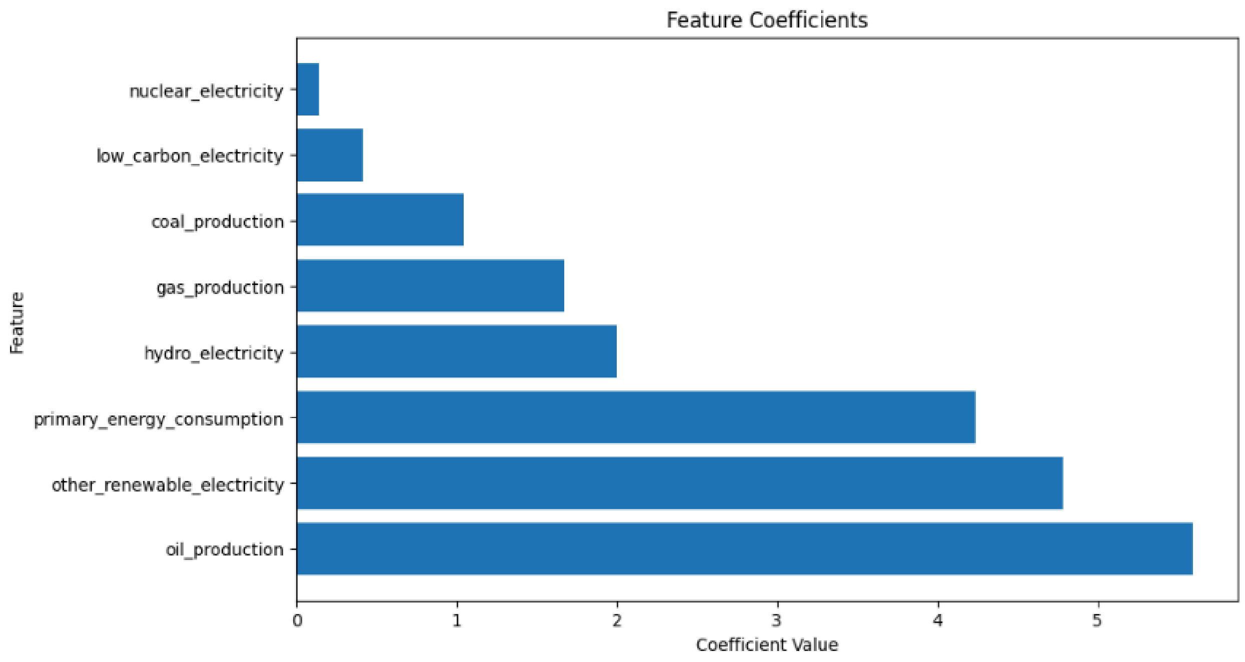


```
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, pred))
print("RMSE:", rmse)
```

```
RMSE: 0.04572984185572557
```

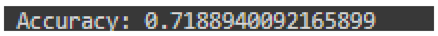
In terms of error, we can clearly say that XGboost is better than linear regression for the target variable GDP.

#x Logistic Regression (In class):
Logistic Regression is like a way to make predictions when you have a bunch of information (these are your independent variables) and want to figure out if something specific will happen or not (that's your outcome, and it can only be one of two things). It estimates the probability that a given input point belongs to a certain class. Logistic regression is suitable for binary classification tasks, such as predicting whether something is true or false, yes or no, high or low, etc.
Logistic regression is suitable for this task because it's a binary classification problem (predicting high or low GDP). The model's coefficients provide insights into the relationship between each feature and the likelihood of a country having a high GDP, making it a valuable tool for understanding the impact of energy consumption and production on economic output.
Visualization:



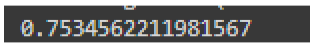
For visualization, we have extracted the coefficients of the logistic regression model (which correspond to the importance of each feature) and created a data frame mapping these coefficients to their respective features then we plotted these coefficients in a bar chart, which helps in understanding the influence of each feature on the model's predictions. From the bar chart we can interpret oil production has the highest importance and nuclear electricity has the lowest importance.

Accuracy of Binary Classification:



```
Accuracy: 0.7188940092165899
```

Model Score:

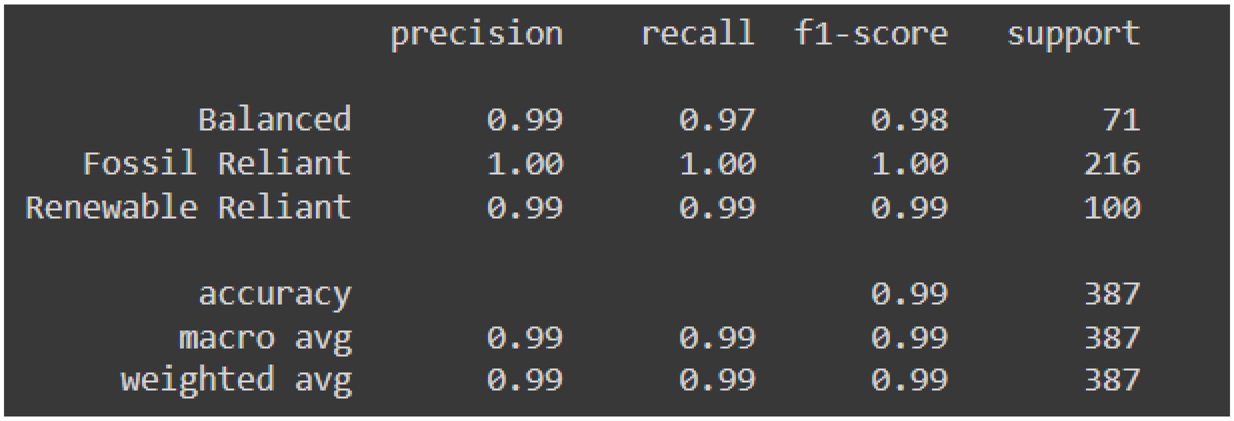


```
0.7534562211981567
```

Random Forest Classifier (Outside Class):

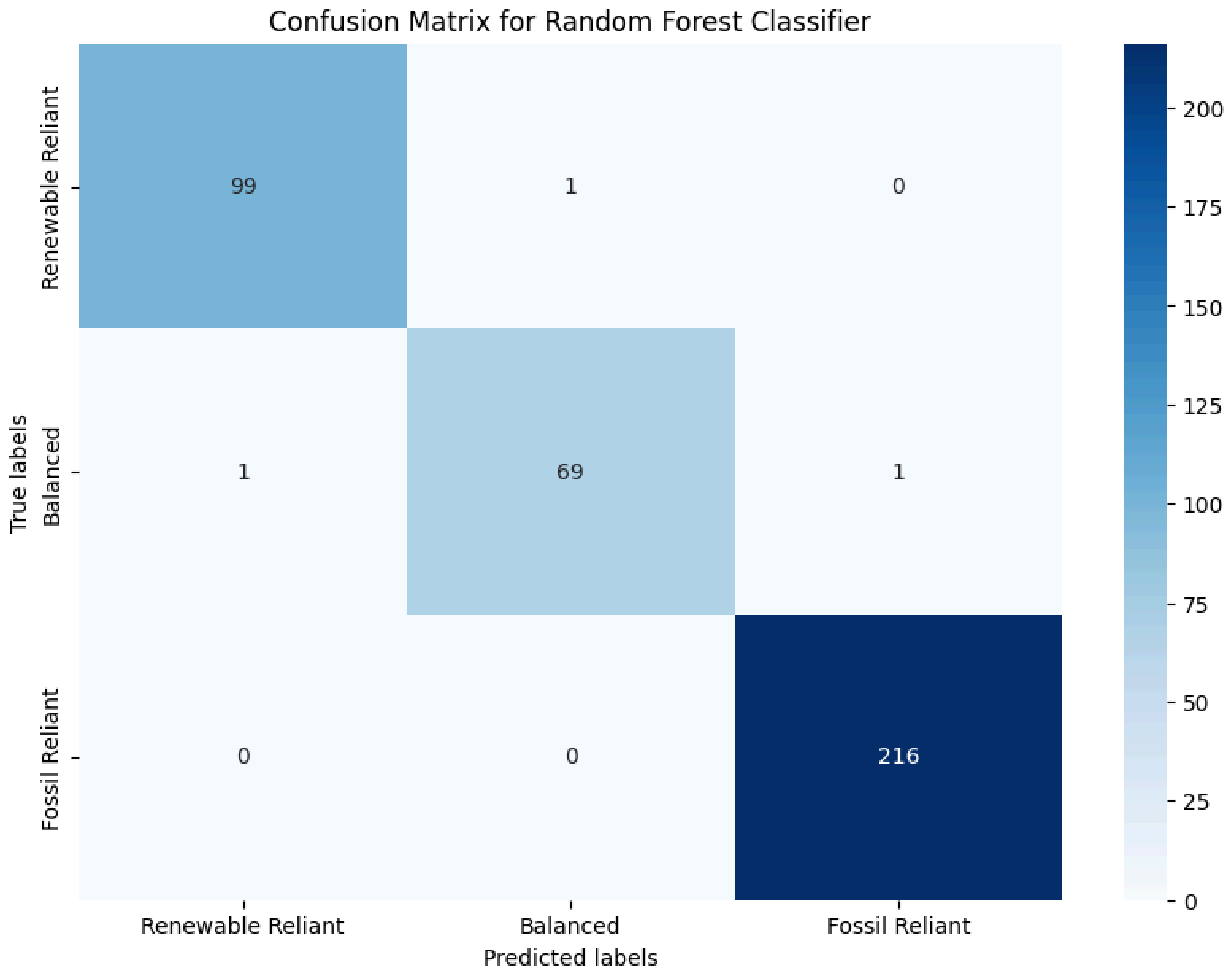
The Random Forest algorithm was chosen for its ability to handle both categorical and continuous input variables and its robustness against overfitting. In this case, the Random Forest model was configured with 100 trees (n_estimators=100), which is a balance between computational efficiency and model performance. The Random Forest model benefits from minimal tuning in many cases due to the ensemble learning, which often performs well with default parameters.

The model's performance was outstanding, as reflected by the precision, recall, and F1-score metrics, all of which are predominantly at or near 1.00. This indicates high accuracy and consistency in the model's predictions. The accuracy score, indicated by the classification report, shows the overall high performance of the model across all classes, with an overall accuracy of 99%.



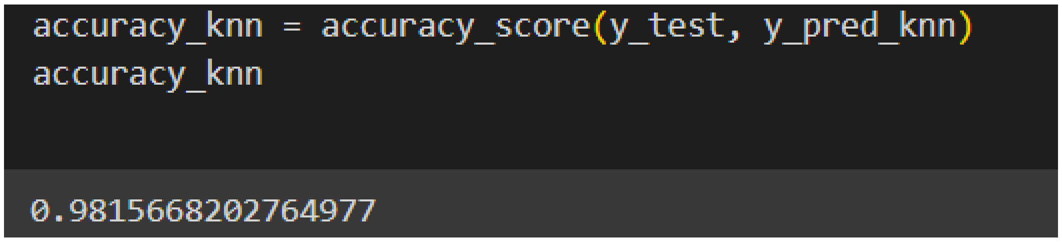
```
                    precision    recall  f1-score   support

          Balanced       0.99      0.97      0.98        71
    Fossil Reliant       1.00      1.00      1.00       216
Renewable Reliant        0.99      0.99      0.99       100

          accuracy                           0.99       387
         macro avg       0.99      0.99      0.99       387
      weighted avg       0.99      0.99      0.99       387
```

Confusion Matrix:



The confusion matrix visualizes the model's performance, with most predictions concentrated along the diagonal, indicating correct classifications. The few off-diagonal elements represent misclassifications, which are very less.

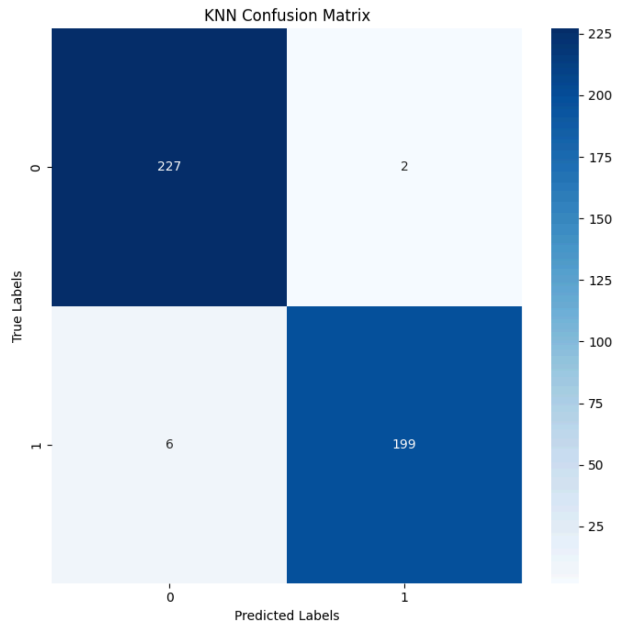
KNN:
The k-Nearest Neighbors algorithm was simple to implement and understand. Given the nature of our dataset, which involves multiple features representing different aspects (e.g., GDP, population, energy metrics), k-NN provides a flexible approach to identifying patterns.
The model was trained using a subset of features from the dataset after preprocessing steps such as feature scaling and missing value imputation. First, we included the countries column but since the continent column was derived from countries the model was getting overfit and the accuracy we obtained was 100%, in addition to that the dimensions were also high due to one hot encoding on countries. With k defined as 5, we trained our model on 80% of the entire dataset that is test data.
The effectiveness of the KNN algorithm was demonstrated through its high accuracy of approximately 98.16% on the test set.



```
accuracy_knn = accuracy_score(y_test, y_pred_knn)
accuracy_knn
```

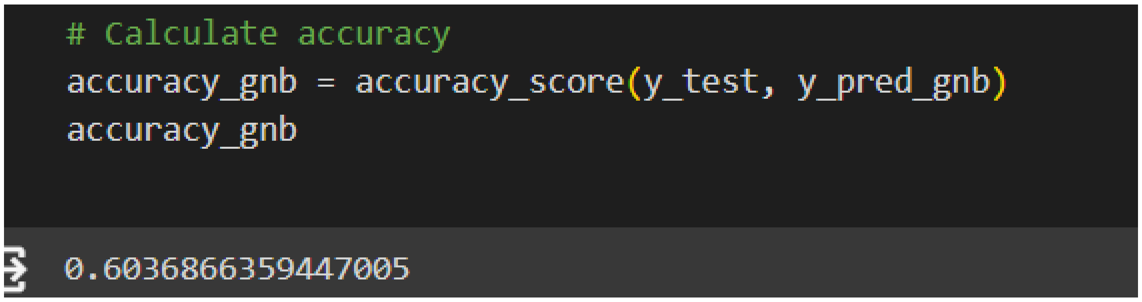
```
0.9815668202764977
```

Confusion Matrix:



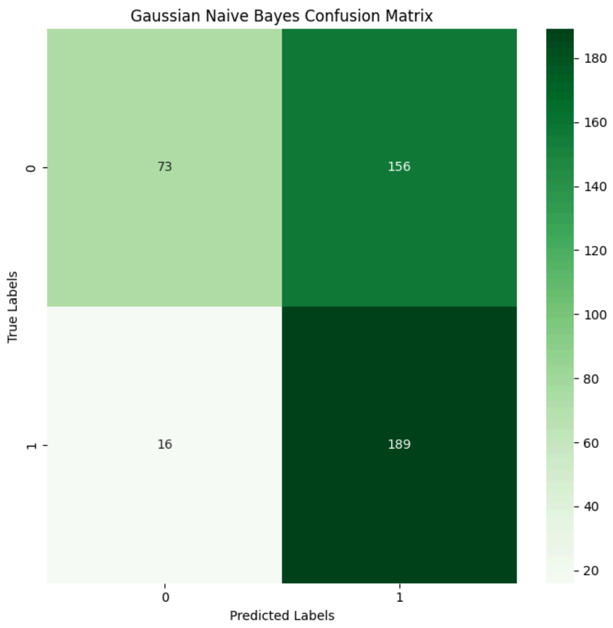
Here as mentioned in the matrix it is performing very well with the high number of true positive and false negative values.
The application of KNN to the dataset revealed the algorithm's strength in handling multi-class classification problems in a multi-dimensional feature space.

Naive Bayes:
Naive Bayes was selected for its efficiency and strong performance in classification tasks involving independent features probabilistic foundation. Given the diverse range of features in the dataset, from GDP to various energy metrics, Naive Bayes offered a straightforward approach to understanding the influence of each feature on the probability of a country belonging to a specific continent.
The initial task was to preprocess the data to ensure compatibility with the algorithm. In naive Bayes, we got an accuracy of 60.73%. The confusion matrix and lower accuracy indicate challenges in handling the multi-class classification with the independence assumption.



```
# Calculate accuracy
accuracy_gnb = accuracy_score(y_test, y_pred_gnb)
accuracy_gnb

0.6036866359447005
```

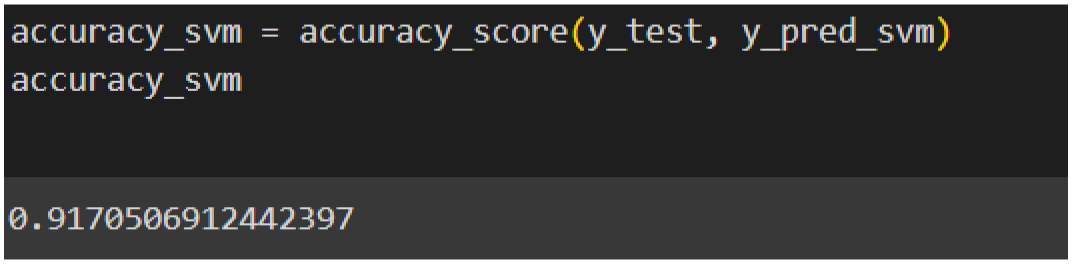
Confusion Matrix:



The number of true negatives and false positives is higher in the naive Bayes.
This states that Naive Bayes doesn't work that well in multi-class, multi-feature classification tasks.

Support Vector Machine(SVM):
We also implemented the SVM algorithm to try out the difference. Since it is used in high-dimensional spaces and its effective in classification problems, We used the Radial Basis Function (RBF) kernel, as we tried with linear but it took much more time to train. We trained our model on the test dataset and then tried that model on the test dataset and got an accuracy of

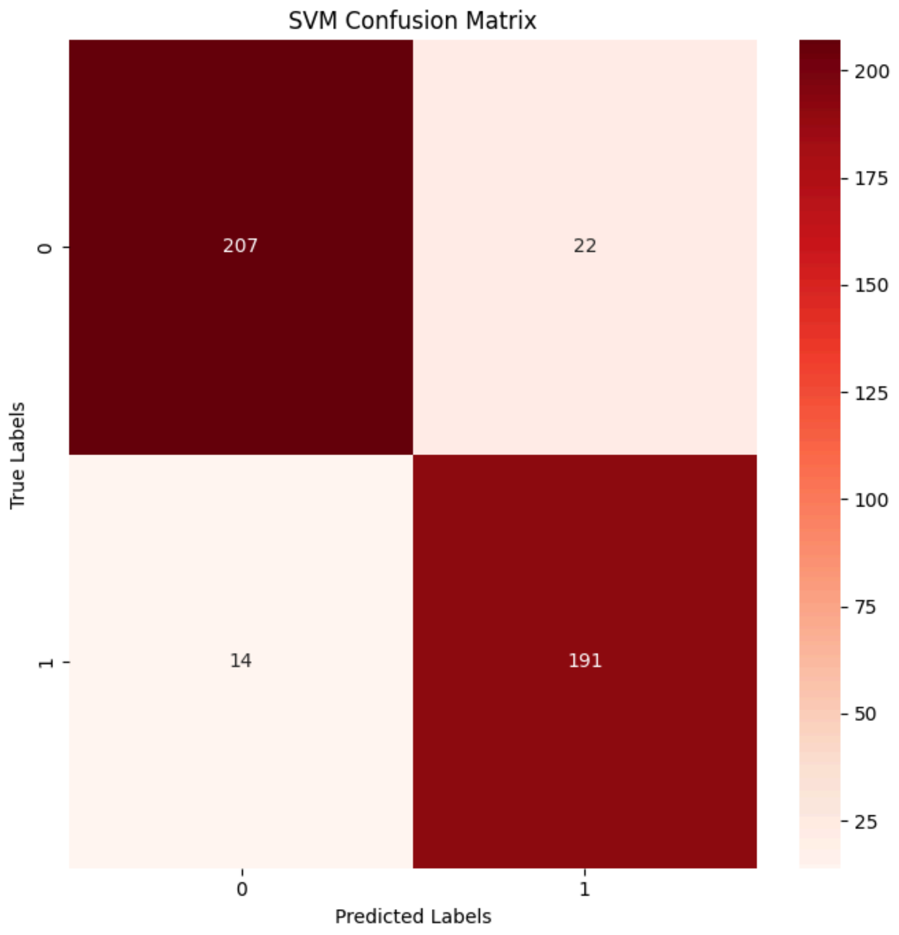
91.70%. Since SVM can show good effectiveness in high dimensional cases it provided better results.



```
accuracy_svm = accuracy_score(y_test, y_pred_svm)
accuracy_svm
```

```
0.9170506912442397
```

The application of the SVM algorithm to the dataset highlights its potential in capturing complex patterns and relationships in high-dimensional data.
Confusion Matrix:



As accuracy indicates the wrong predicted values are very low in SVM compared to Naive Bayes. Accuracy was not as good as KNN but it demonstrates SVM's robustness in dealing with complex, high-dimensional data.
Hence in the classification task between KNN, Naive Bayes, and SVM, KNN suits best our dataset.

References:

- https://www.dataquest.io/blog/learning-curves-machine-learning/
- https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/#:~:text=Logistic%20regression%20is%20a%20supervised%20machine%20learning%20algorithm%20that%20accomplishes,1%2C%20or%20true%2Ffalse.
- https://xgboost.readthedocs.io/en/stable/tutorials/model.html
- https://stackoverflow.com/
- https://www.analyticsvidhya.com/
- https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/#:~:text=The%20K%2DNearest%20Neighbors%20(KNN)%20algorithm%20is%20a%20popular,have%20similar%20labels%20or%20values.
- https://www.geeksforgeeks.org/naive-bayes-classifiers/
- https://www.geeksforgeeks.org/support-vector-machine-algorithm/
- https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/