Project id: cdsds561-project-1
Bucket Name: hw2nirbhgsutil
Bucket-2 Name:hw4nirbhgsutil
Directory Name: test-dir
Github Link:CDS-DS-561-hw2/hw6 at master · nirbhay221/CDS-DS-561-hw2 (github.com)
Topic name : projects/cdsds561-project-1/topics/hw3nirbhgsutil
Topic Id: hw3nirbhgsutil
Subscription Id: hw3nirbhgsutil-sub
Service Account email for pub sub:
pubsubserviceacc-hw3-nirbh@cdsds561-project-1.iam.gserviceaccount.com
Service Account for pub sub: pubsubServiceAcc-hw3-nirbh
Service Account for SQL:
cloud-sql-authorize-vm@cdsds561-project-1.iam.gserviceaccount.com
Reserved IP : 34.75.102.252
Sql database instance: my-database
Sql database: First-Trial , Second-Trial
Sql Tables : Clients, main_table, error_logs.
Vm-instance : 5 - directory - model-1 - main.py , main-1.py , 6 - main.py , main-1.py

If you are starting on a new vm, you should install the following libraries:

pip install apache-beam;
pip install pandas;
pip install scikit-learn;

First model uses client IP for predicting the country from which the request originated. I used a Decision Tree Classifier and a loop for the random states for determining the best state possible that will provide the best test accuracy. Other than this Random Forest Classifier was also used for determining if a better accuracy can be reached using the following classifier model but it takes a huge amount of time for determining the test accuracy for the various random states. Following first model is on instance 5, works well on the directory : model - 1 - main.py represents the Random Forest Classifier and main-1.py represents the Decision Tree Classifier.

When i use vm-instance 5, I get the following result for the Decision Tree Classifier:

```
-bash: python: command not found
nmalhotr@instance-5:~/model-1$ python3 main-1.py
100000
       country       client_ip
0      Ireland   234.203.57.38
1      Vietnam  244.65.179.157
2      Tunisia  47.163.244.142
3  New Zealand  14.165.203.121
4      Iceland   72.223.10.166
Random State 0: Model: DecisionTreeClassifier()
Random State 0: Testing Accuracy: 99.66%
Random State 1: Model: DecisionTreeClassifier()
Random State 1: Testing Accuracy: 99.75%
Random State 2: Model: DecisionTreeClassifier()
Random State 2: Testing Accuracy: 99.66%
Random State 3: Model: DecisionTreeClassifier()
Random State 3: Testing Accuracy: 99.57%
Random State 4: Model: DecisionTreeClassifier()
Random State 4: Testing Accuracy: 99.65%
Random State 5: Model: DecisionTreeClassifier()
Random State 5: Testing Accuracy: 99.69%
Random State 6: Model: DecisionTreeClassifier()
Random State 6: Testing Accuracy: 99.57%
Random State 7: Model: DecisionTreeClassifier()
Random State 7: Testing Accuracy: 99.66%
Random State 8: Model: DecisionTreeClassifier()
Random State 8: Testing Accuracy: 99.60%
Random State 9: Model: DecisionTreeClassifier()
Random State 9: Testing Accuracy: 99.67%
Random State 10: Model: DecisionTreeClassifier()
Random State 10: Testing Accuracy: 99.61%
Random State 11: Model: DecisionTreeClassifier()
Random State 11: Testing Accuracy: 99.63%
Random State 12: Model: DecisionTreeClassifier()
```

```
Random State 82: Testing Accuracy: 99.65%
Random State 83: Model: DecisionTreeClassifier()
Random State 83: Testing Accuracy: 99.81%
Random State 84: Model: DecisionTreeClassifier()
Random State 84: Testing Accuracy: 99.78%
Random State 85: Model: DecisionTreeClassifier()
Random State 85: Testing Accuracy: 99.59%
Random State 86: Model: DecisionTreeClassifier()
Random State 86: Testing Accuracy: 99.54%
Random State 87: Model: DecisionTreeClassifier()
Random State 87: Testing Accuracy: 99.59%
Random State 88: Model: DecisionTreeClassifier()
Random State 88: Testing Accuracy: 99.49%
Random State 89: Model: DecisionTreeClassifier()
Random State 89: Testing Accuracy: 99.63%
Random State 90: Model: DecisionTreeClassifier()
Random State 90: Testing Accuracy: 99.61%
Random State 91: Model: DecisionTreeClassifier()
Random State 91: Testing Accuracy: 99.65%
Random State 92: Model: DecisionTreeClassifier()
Random State 92: Testing Accuracy: 99.67%
Random State 93: Model: DecisionTreeClassifier()
Random State 93: Testing Accuracy: 99.53%
Random State 94: Model: DecisionTreeClassifier()
Random State 94: Testing Accuracy: 99.66%
Random State 95: Model: DecisionTreeClassifier()
Random State 95: Testing Accuracy: 99.62%
Random State 96: Model: DecisionTreeClassifier()
Random State 96: Testing Accuracy: 99.76%
Random State 97: Model: DecisionTreeClassifier()
Random State 97: Testing Accuracy: 99.67%
Random State 98: Model: DecisionTreeClassifier()
Random State 98: Testing Accuracy: 99.67%
Random State 99: Model: DecisionTreeClassifier()
Random State 99: Testing Accuracy: 99.68%
Best Random State: 83
Best Testing Accuracy: 99.81
```

I get the following test accuracy of 99.81 % when I run the decision tree classifier model on vm - instance 5.

When i use vm-instance 5, I get the following result for the Random Forest Classifier:

```
nmalhotr@instance-5:~/model-1$ python3 main.py
       country       client_ip
0      Ireland   234.203.57.38
1      Vietnam  244.65.179.157
2      Tunisia  47.163.244.142
3  New Zealand  14.165.203.121
4      Iceland   72.223.10.166
Random State 81: Model: RandomForestClassifier(n_estimators=500)
Random State 81: Testing Accuracy: 99.61%
Best Random State: 81
Best Testing Accuracy: 99.61
nmalhotr@instance-5:~/model-1$
```

I get the following test accuracy of 99.61 % when I run the random forest classifier model on vm - instance 5.

Decision Trees are highly interpretable and they provide a clear decision path, making it easier to understand and determine why the model makes specific predictions. This can be crucial for gaining insights into data and finding relationships. Decision trees can handle the categorical data naturally so I think that's why it works well for our categorical columns. Also, I used label encoding for the columns (IPs and country) for the preprocessing steps. Decision trees can capture deterministic connection between the IPs and the countries and it can also capture the non-linear relationships in the data. It provides a feature importance score for identifying the most influential features in my dataset. It is relatively fast to train and for experimentation, It also allows for controlling tree depth for mitigating overfitting. Also when I increase the dataset, we get higher accuracy due to deterministic nature between the variables and the target variable.

Random Forest Classifier is an ensemble learning method that combines multiple decision trees for making predictions which often provides high predictive accuracy, which is valuable for tasks where accurate predictions are essential. It is less prone to overfitting compared to individual decision trees and it achieves this by averaging predictions from multiple trees which reduces the risk of learning noise in the data. Also it captures the complex interactions between features like age, client_ip, and country. By combining the predictions from multiple trees, random forest reduces the risk of model bias making it suitable for datasets with imbalanced classes. I didn't have to fine-tune hyperparameters extensively for achieving good results.

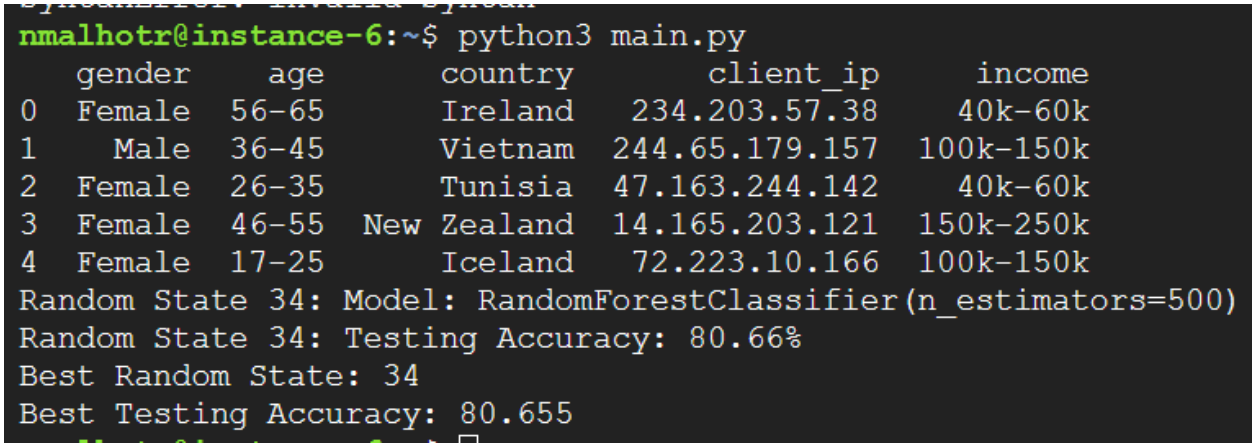
I used a random state of 123 for making the Second-Trial's main_table and the errror_logs.

The second model uses age, country, and client_ip to predict income. For the following too, I used a random forest classifier and the decision tree classifier for achieving the accuracy above 80 %. I used a decision tree classifier for determining the appropriate random state and then I used a random forest classifier for determining the best test accuracy. I ran the following decision tree classifier model on vm-instance-6, I get the following output:

```
nmalhotr@instance-6:~$ python3 main-1.py
   gender    age      country       client_ip     income
0  Female  56-65      Ireland   234.203.57.38    40k-60k
1    Male  36-45      Vietnam  244.65.179.157  100k-150k
2  Female  26-35      Tunisia  47.163.244.142    40k-60k
3  Female  46-55  New Zealand  14.165.203.121  150k-250k
4  Female  17-25      Iceland   72.223.10.166  100k-150k
Random State 0: Model: DecisionTreeClassifier()
Random State 0: Testing Accuracy: 80.26%
Random State 1: Model: DecisionTreeClassifier()
Random State 1: Testing Accuracy: 80.31%
Random State 2: Model: DecisionTreeClassifier()
Random State 2: Testing Accuracy: 79.98%
Random State 3: Model: DecisionTreeClassifier()
Random State 3: Testing Accuracy: 80.86%
Random State 4: Model: DecisionTreeClassifier()
Random State 4: Testing Accuracy: 80.67%
Random State 5: Model: DecisionTreeClassifier()
Random State 5: Testing Accuracy: 80.42%
Random State 6: Model: DecisionTreeClassifier()
Random State 6: Testing Accuracy: 80.45%
Random State 7: Model: DecisionTreeClassifier()
Random State 7: Testing Accuracy: 81.02%
Random State 8: Model: DecisionTreeClassifier()
Random State 8: Testing Accuracy: 80.05%
Random State 9: Model: DecisionTreeClassifier()
Random State 9: Testing Accuracy: 80.11%
Random State 10: Model: DecisionTreeClassifier()
Random State 10: Testing Accuracy: 80.77%
Random State 11: Model: DecisionTreeClassifier()
Random State 11: Testing Accuracy: 79.79%
Random State 12: Model: DecisionTreeClassifier()
Random State 12: Testing Accuracy: 80.27%
Random State 13: Model: DecisionTreeClassifier()
Random State 13: Testing Accuracy: 80.88%
```

```
Random State 82: Testing Accuracy: 80.25%
Random State 83: Model: DecisionTreeClassifier()
Random State 83: Testing Accuracy: 80.93%
Random State 84: Model: DecisionTreeClassifier()
Random State 84: Testing Accuracy: 80.72%
Random State 85: Model: DecisionTreeClassifier()
Random State 85: Testing Accuracy: 80.70%
Random State 86: Model: DecisionTreeClassifier()
Random State 86: Testing Accuracy: 80.52%
Random State 87: Model: DecisionTreeClassifier()
Random State 87: Testing Accuracy: 80.79%
Random State 88: Model: DecisionTreeClassifier()
Random State 88: Testing Accuracy: 80.98%
Random State 89: Model: DecisionTreeClassifier()
Random State 89: Testing Accuracy: 80.05%
Random State 90: Model: DecisionTreeClassifier()
Random State 90: Testing Accuracy: 80.44%
Random State 91: Model: DecisionTreeClassifier()
Random State 91: Testing Accuracy: 80.19%
Random State 92: Model: DecisionTreeClassifier()
Random State 92: Testing Accuracy: 80.68%
Random State 93: Model: DecisionTreeClassifier()
Random State 93: Testing Accuracy: 80.30%
Random State 94: Model: DecisionTreeClassifier()
Random State 94: Testing Accuracy: 80.88%
Random State 95: Model: DecisionTreeClassifier()
Random State 95: Testing Accuracy: 80.47%
Random State 96: Model: DecisionTreeClassifier()
Random State 96: Testing Accuracy: 80.77%
Random State 97: Model: DecisionTreeClassifier()
Random State 97: Testing Accuracy: 80.38%
Random State 98: Model: DecisionTreeClassifier()
Random State 98: Testing Accuracy: 80.23%
Random State 99: Model: DecisionTreeClassifier()
Random State 99: Testing Accuracy: 79.83%
Best Random State: 21
Best Testing Accuracy: 81.03
```

The best accuracy I get for the following decision tree model is 81.03 for random state 21. I ran the following random forest classifier model on vm-instance-6, I get the following output:



Decision Trees are highly interpretable and they provide a clear decision path, making it easier to understand and determine why the model makes specific predictions. This can be crucial for gaining insights into data and finding relationships. Decision trees can handle the categorical data naturally so I think that's why it works well for our categorical columns. Also, I label encoding for the columns for the preprocessing steps. Decision trees can capture deterministic connection between the (IPs, countries, age groups) and income groups and it can also capture the non-linear relationships in the data. It provides a feature importance score for identifying the most influential features in my dataset. It is relatively fast to train and for experimentation, It also allows for controlling tree depth for mitigating overfitting. I had to find the columns that would increase the test accuracy like I already knew the deterministic nature between IPs and countries so I tried to find other columns that had relationships between them so I tried and tested a few of them and finally got the test accuracy above 80 %. Also when I increase the dataset, we get higher accuracy due to deterministic nature between the variables and the target variable.

Random Forest Classifier is an ensemble learning method that combines multiple decision trees for making predictions which often provides high predictive accuracy, which is valuable for tasks where accurate predictions are essential. It is less prone to overfitting compared to individual decision trees and it achieves this by averaging predictions from multiple trees which reduces the risk of learning noise in the data. Also it captures the complex interactions between features like age, client_ip, and country. By combining the predictions from multiple trees, random forest reduces the risk of model bias making it suitable for datasets with imbalanced classes. I didn't had to fine-tune hyperparameters extensively for achieving good results.