Credit Card Fault Detection

Model Development and Deployment Process

Date	09-01-2024
Prepared by	Makineedi Venkat Dinesh

Fraud Prediction

Document details the process of estimating fault transaction based on machine learning approach.

Author	Makineedi Venkat Dinesh	Date – 09/01/2024

Table of Content

Credit Card Fault Detection Report

1)Prediction Approach

I experimented with three approaches and finalized one approach giving best area under ROC curve.

Approach 1: Normalizing 'Amount' column and eliminating 'time' column.

Approach 2: Normalizing all features.

Approach 3: Resampling all the data to get balanced dataset.

In every approach I trained three models:

Model1: lightGBM -> hyperparameter tuning is done for this model.

Model2: Logistic Regression

Model3: KNeighbours classification

Approach 1: Normalizing 'Amount' column and eliminating 'time' column

Model1: lightGBM reports:

TRAIN MODEL CLASSIFICATION REPORT

---

	precision	recall	f1-score	support
No Fraud	1.00	1.00	1.00	227451
Fraud	1.00	1.00	1.00	394
accuracy			1.00	227845

TEST MODEL CLASSIFICATION REPORT

---

	precision	recall	f1-score	support
No Fraud	1.00	1.00	1.00	56864
Fraud	0.98	0.81	0.88	98
accuracy			1.00	56962


Model performance report
lgbm_clf_tuned
Confusion Matrix
1 (Fraud) actual
0 (No Fraud) actual
19
79
56862
2
0 (No Fraud) pred
1 (Fraud) pred
Metrics
F1_score
Recall
Precision
Accuracy
0.8827
0.8061
0.9753
0.9996
0
0.5
1
ROC curve (0.98)
true positive rate
false positive rate
Precision - Recall curve
precision
recall
Cumulative gains curve
Percentage positive targeted
Percentage contacted
0.001
0.002
0.003
0.004
0.005
0.006
Feature importance
V22
V26
V13
V12
V24
V15
V3
Amount
V25
V1
V4
V7
V19
V14
V11
V9
V23
V28
V21
V16
V27
V6
V5
V10
V2
V20
V18
V17
V8

Model2: Logistic Regression reports:

TRAIN MODEL CLASSIFICATION REPORT

	precision	recall	f1-score	support
No Fraud	1.00	1.00	1.00	227451
Fraud	0.87	0.62	0.72	394
accuracy			1.00	227845

TEST MODEL CLASSIFICATION REPORT

	precision	recall	f1-score	support
No Fraud	1.00	1.00	1.00	56864
Fraud	0.92	0.61	0.74	98
accuracy			1.00	56962


Model performance report
lr
Confusion Matrix
1 (Fraud) actual
0 (No Fraud) actual
38
60
56859
5
0 (No Fraud) pred
1 (Fraud) pred
Metrics
F1_score
0.7362
Recall
0.6122
Precision
0.9231
Accuracy
0.9992
ROC curve (0.98)
true positive rate
false positive rate
Precision - Recall curve
precision
recall
Cumulative gains curve
Percentage positive targeted
Percentage contacted

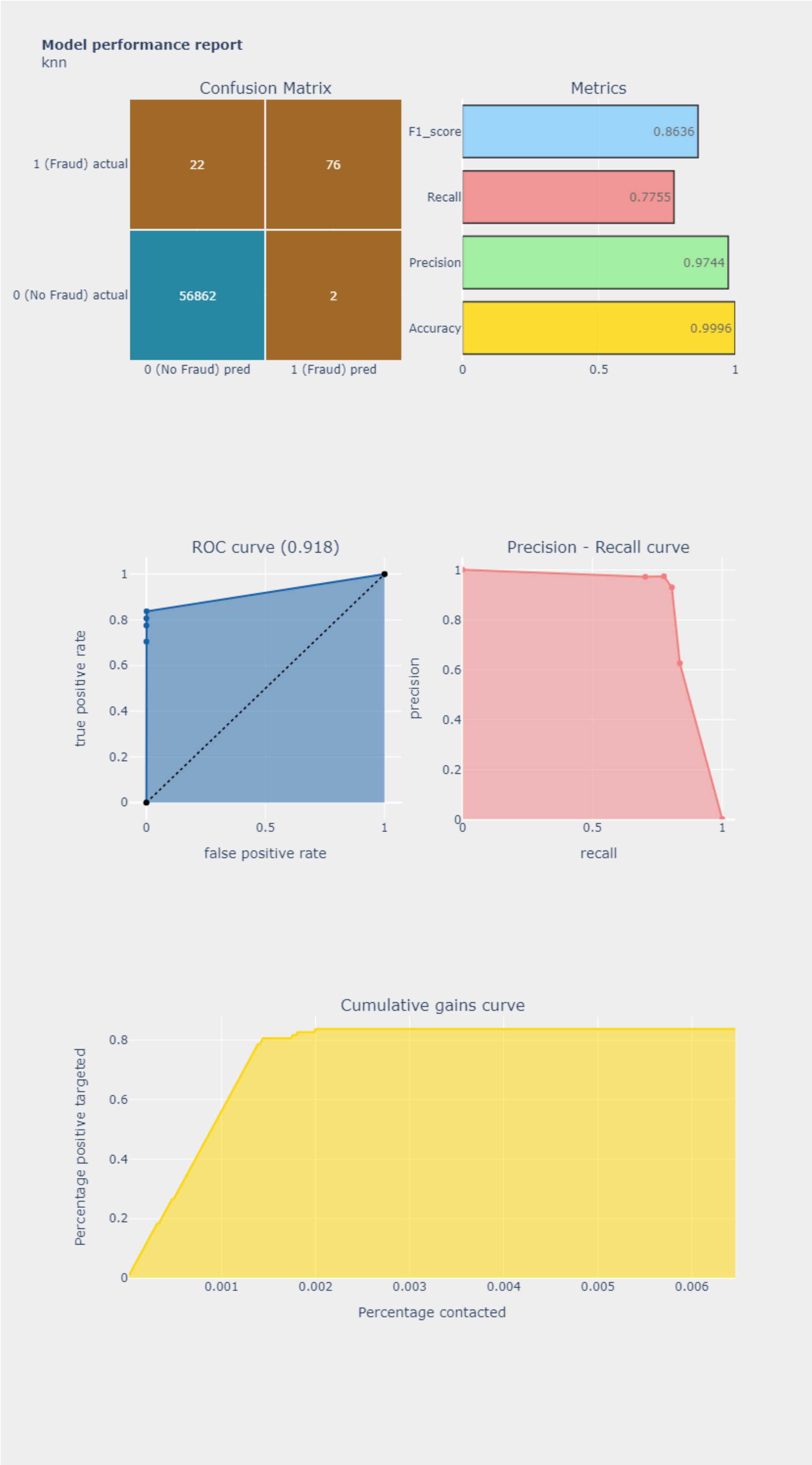
Model3: KNeighbors Classification reports:

TRAIN MODEL CLASSIFICATION REPORT

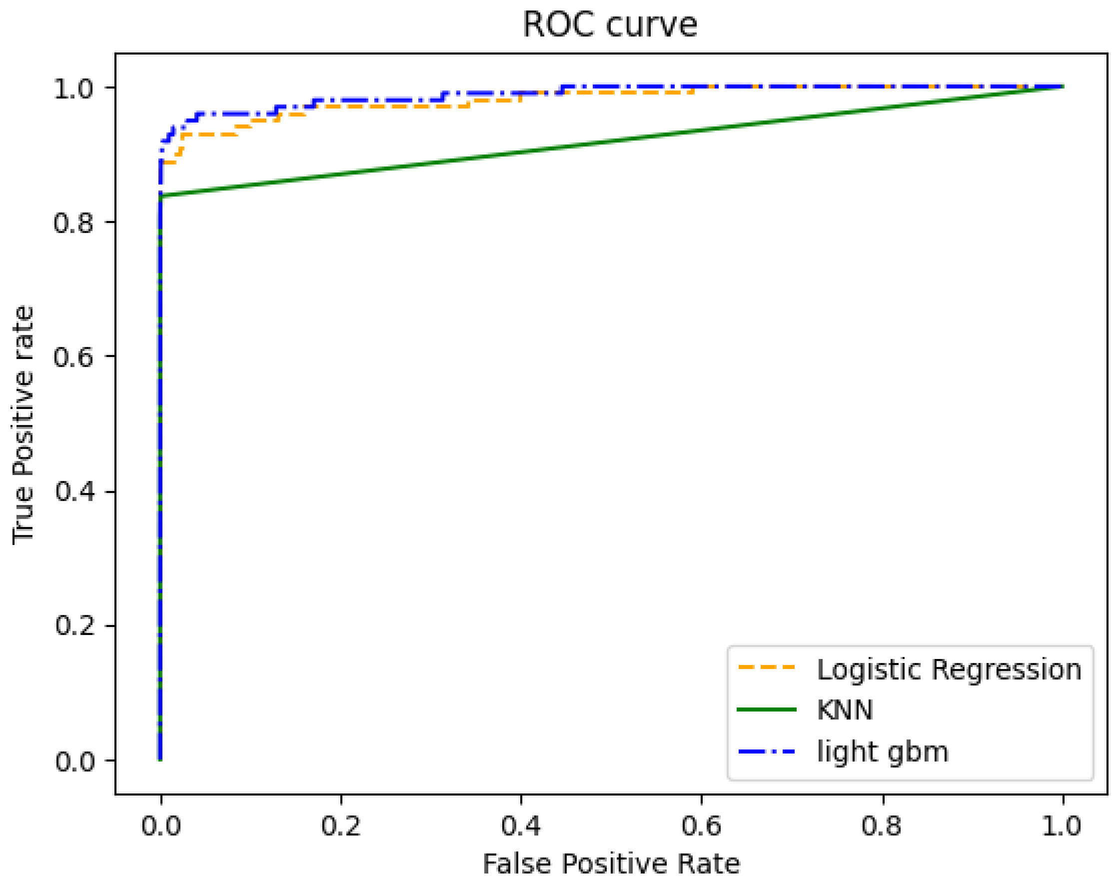
	precision	recall	f1-score	support
No Fraud	1.00	1.00	1.00	227451
Fraud	0.98	0.79	0.87	394
accuracy			1.00	227845

TEST MODEL CLASSIFICATION REPORT

	precision	recall	f1-score	support
No Fraud	1.00	1.00	1.00	56864
Fraud	0.97	0.78	0.86	98
accuracy			1.00	56962


Model performance report
knn
Confusion Matrix
1 (Fraud) actual
0 (No Fraud) actual
0 (No Fraud) pred
1 (Fraud) pred
22
76
56862
2
Metrics
F1_score
Recall
Precision
Accuracy
0.8636
0.7755
0.9744
0.9996
ROC curve (0.918)
true positive rate
false positive rate
Precision - Recall curve
precision
recall
Cumulative gains curve
Percentage positive targeted
Percentage contacted

ROC Curve for lightgbm, Logistic Regression and K-Neighbors Models:



From the above reports and ROC curve, lightGBM model is having highest ROC area of 0.98.

Approach 2: Normalizing all features.

Model1: lightGBM reports:

TRAIN MODEL CLASSIFICATION REPORT

---

	precision	recall	f1-score	support
No Fraud	1.00	1.00	1.00	227468
Fraud	1.00	1.00	1.00	377
accuracy			1.00	227845

TEST MODEL CLASSIFICATION REPORT

---

	precision	recall	f1-score	support
No Fraud	1.00	1.00	1.00	56847
Fraud	0.93	0.83	0.88	115
accuracy			1.00	56962

Model performance report
lgbm_clf_tuned
Confusion Matrix
1 (Fraud) actual
0 (No Fraud) actual
0 (No Fraud) pred
1 (Fraud) pred
20
95
56840
7
Metrics
F1_score
Recall
Precision
Accuracy
0.8756
0.8261
0.9314
0.9995
0
0.5
1
ROC curve (0.987)
true positive rate
false positive rate
Precision - Recall curve
precision
recall
Cumulative gains curve
Percentage positive targeted
Percentage contacted
0.001
0.002
0.003
0.004
0.005
0.006
Feature importance
0
200
400
600
800
V26
Time
Amount
V22
V13
V11
V24
V15
V12
V4
V19
V9
V3
V16
V27
V21
V14
V1
V28
V10
V25
V18
V5
V17
V7
V20
V6
V23
V2
V8

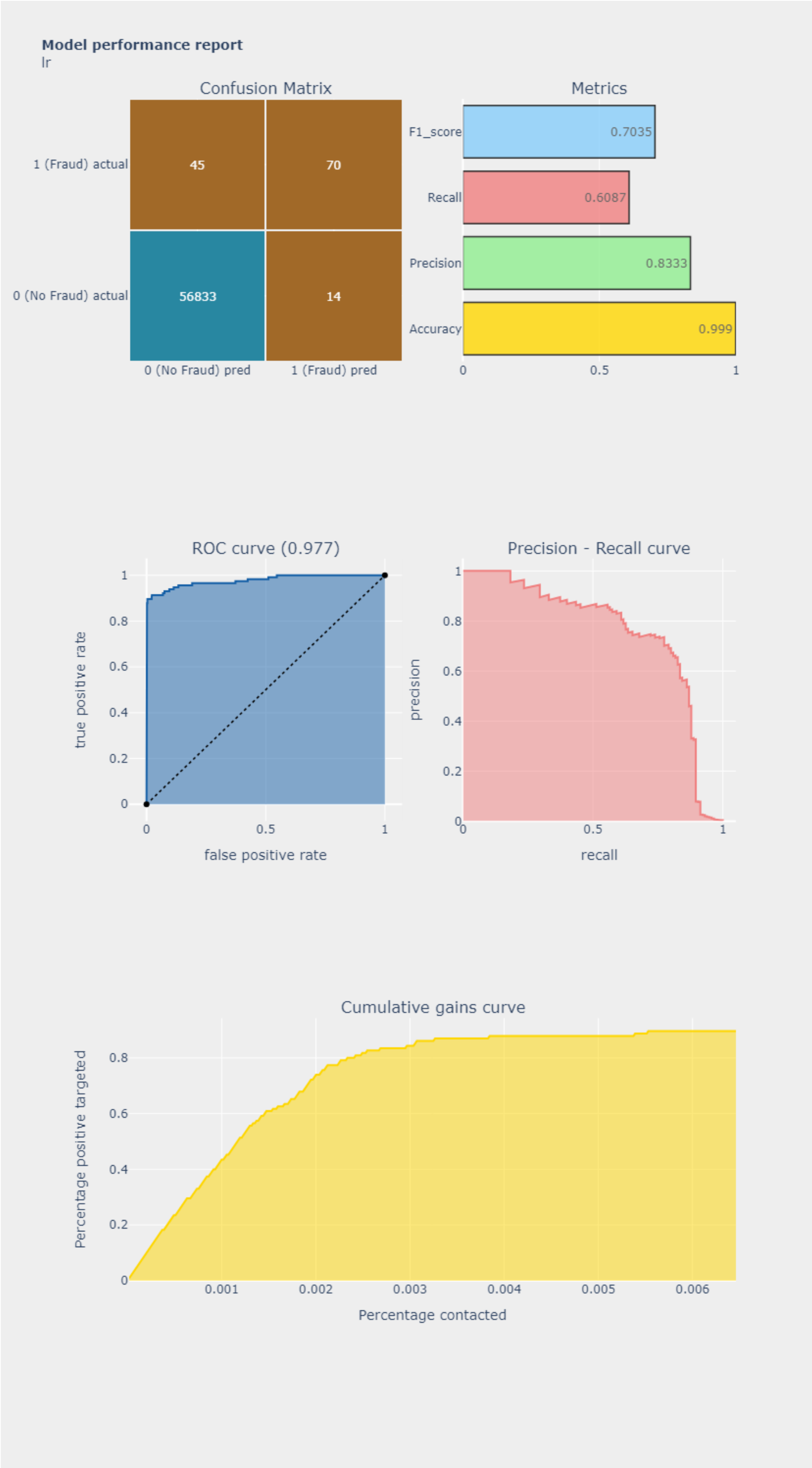
Model2: Logistic Regression reports:

TRAIN MODEL CLASSIFICATION REPORT

	precision	recall	f1-score	support
No Fraud	1.00	1.00	1.00	227468
Fraud	0.89	0.63	0.74	377
accuracy			1.00	227845

TEST MODEL CLASSIFICATION REPORT

	precision	recall	f1-score	support
No Fraud	1.00	1.00	1.00	56847
Fraud	0.83	0.61	0.70	115
accuracy			1.00	56962


Model performance report
lr
Confusion Matrix
1 (Fraud) actual
0 (No Fraud) actual
45
70
56833
14
0 (No Fraud) pred
1 (Fraud) pred
Metrics
F1_score
Recall
Precision
Accuracy
0.7035
0.6087
0.8333
0.999
0
0.5
1
ROC curve (0.977)
true positive rate
false positive rate
Precision - Recall curve
precision
recall
Cumulative gains curve
Percentage positive targeted
Percentage contacted
0.001
0.002
0.003
0.004
0.005
0.006

Model3: KNeighbors Classification reports:

TRAIN MODEL CLASSIFICATION REPORT

---

	precision	recall	f1-score	support
No Fraud	1.00	1.00	1.00	227468
Fraud	0.97	0.79	0.87	377
accuracy			1.00	227845

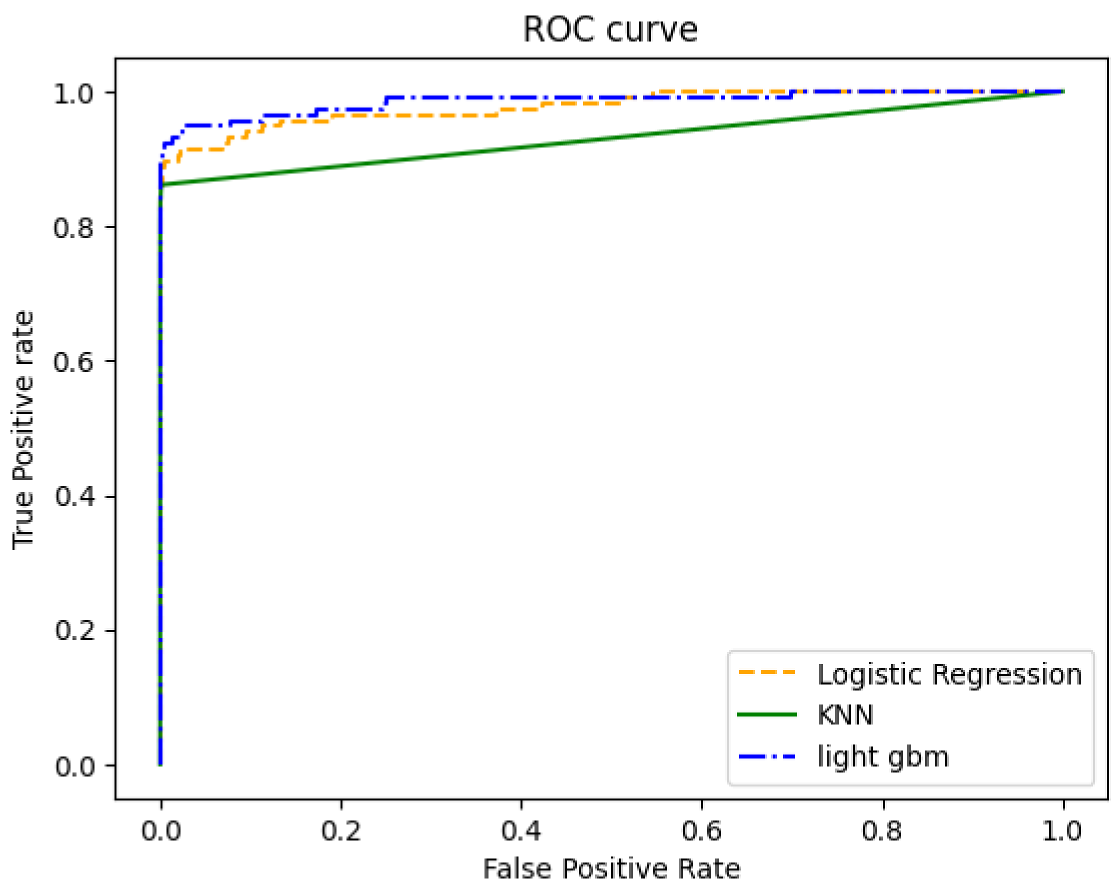
TEST MODEL CLASSIFICATION REPORT

---

	precision	recall	f1-score	support
No Fraud	1.00	1.00	1.00	56847
Fraud	0.95	0.78	0.86	115
accuracy			1.00	56962


Model performance report
knn
Confusion Matrix
1 (Fraud) actual
0 (No Fraud) actual
25
90
56842
5
0 (No Fraud) pred
1 (Fraud) pred
Metrics
F1_score
Recall
Precision
Accuracy
0.8571
0.7826
0.9474
0.9995
0
0.5
1
ROC curve (0.93)
true positive rate
false positive rate
Precision - Recall curve
precision
recall
Cumulative gains curve
Percentage positive targeted
Percentage contacted
0.001
0.002
0.003
0.004
0.005
0.006

ROC Curve for lightgbm, Logistic Regression and K-Neighbors Models:



Approach 3: Resampling.

Model1: lightGBM reports:

TRAIN MODEL CLASSIFICATION REPORT

---

	precision	recall	f1-score	support
No Fraud	0.93	0.98	0.96	383
Fraud	0.98	0.93	0.96	404
accuracy			0.96	787

TEST MODEL CLASSIFICATION REPORT

---

	precision	recall	f1-score	support
No Fraud	0.93	0.94	0.94	109
Fraud	0.93	0.91	0.92	88
accuracy			0.93	197


Model performance report
lgbm_clf_tuned_b
Confusion Matrix
1 (Fraud) actual
0 (No Fraud) actual
0 (No Fraud) pred
1 (Fraud) pred
8
80
103
6
Metrics
F1_score
Recall
Precision
Accuracy
0.9195
0.9091
0.9302
0.9289
ROC curve (0.972)
true positive rate
false positive rate
Precision - Recall curve
precision
recall
Cumulative gains curve
Percentage positive targeted
Percentage contacted
Feature importance
V4
V14
V12
V10
V8
V19
V11
V21
V1
V17
V13
V16
V6
V9
V18
V7
V3
Amount
V22
V27
V23
V24
Time
V5
V2

Model2: Logistic Regression reports:

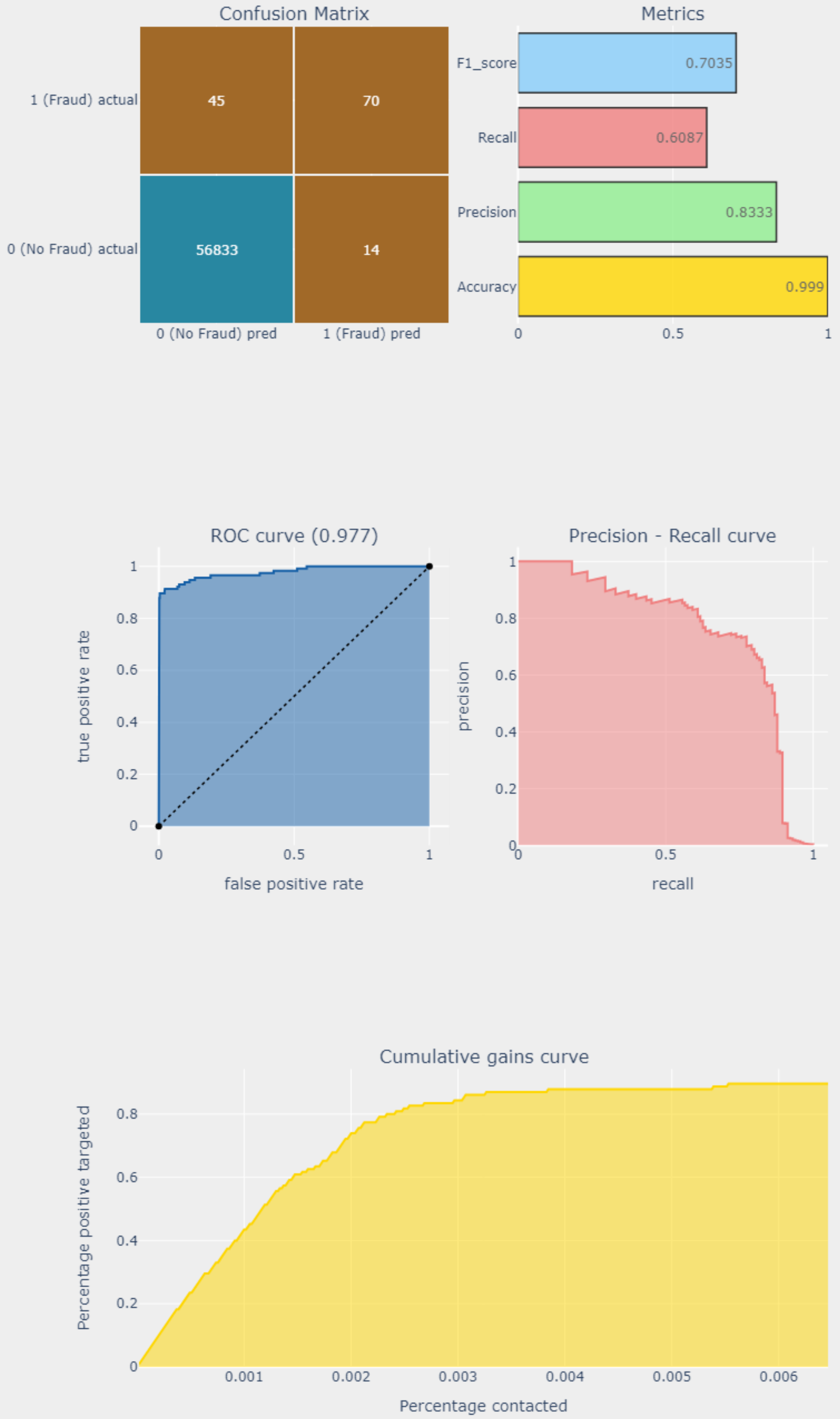
TRAIN MODEL CLASSIFICATION REPORT

	precision	recall	f1-score	support
No Fraud	0.92	0.98	0.95	383
Fraud	0.98	0.92	0.95	404
accuracy			0.95	787

TEST MODEL CLASSIFICATION REPORT

	precision	recall	f1-score	support
No Fraud	0.94	0.97	0.95	109
Fraud	0.96	0.92	0.94	88
accuracy			0.95	197

Model performance report
lr



Model3: KNeighbors Classification reports:

TRAIN MODEL CLASSIFICATION REPORT

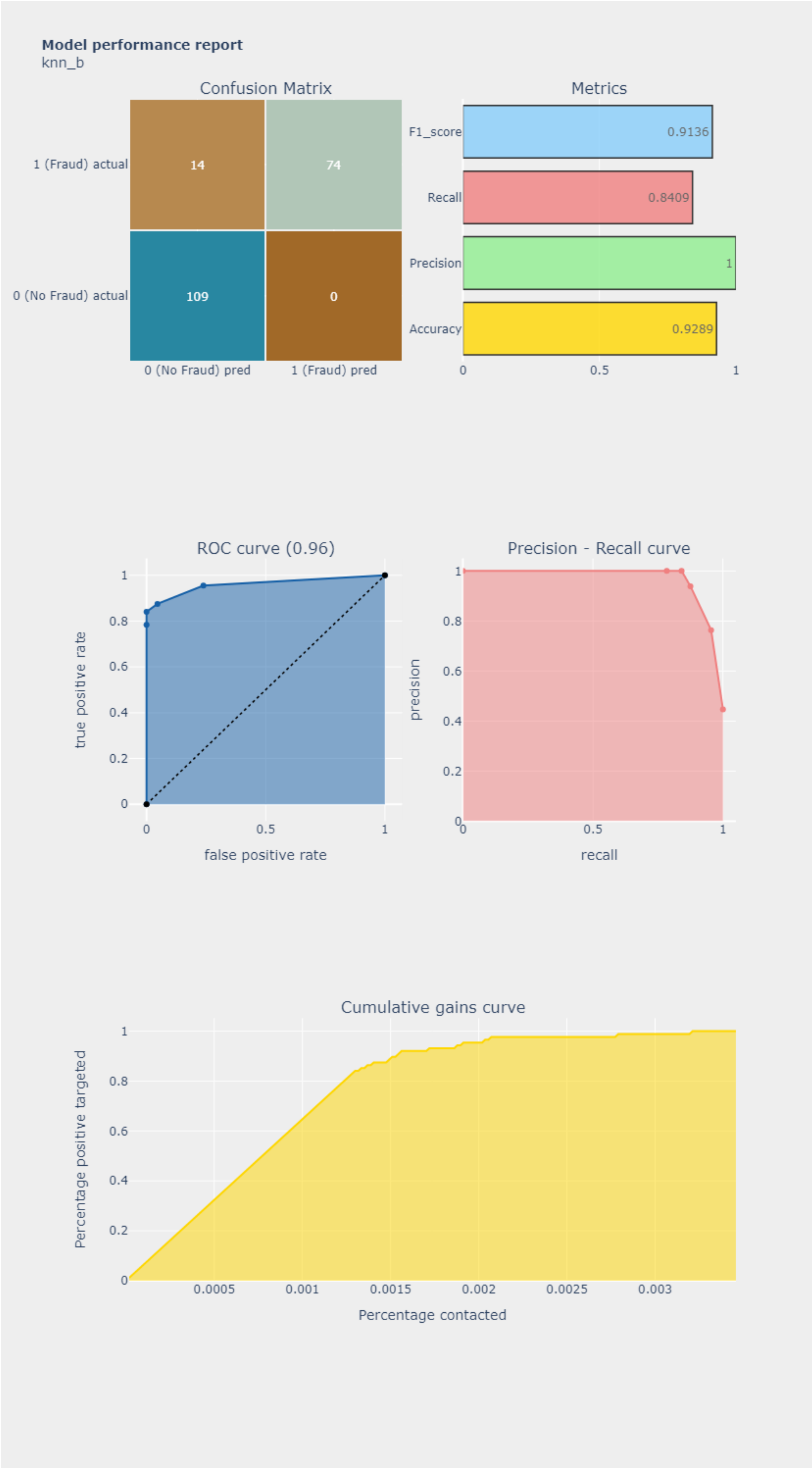
---

	precision	recall	f1-score	support
No Fraud	0.88	0.99	0.93	383
Fraud	0.99	0.87	0.93	404
accuracy			0.93	787

TEST MODEL CLASSIFICATION REPORT

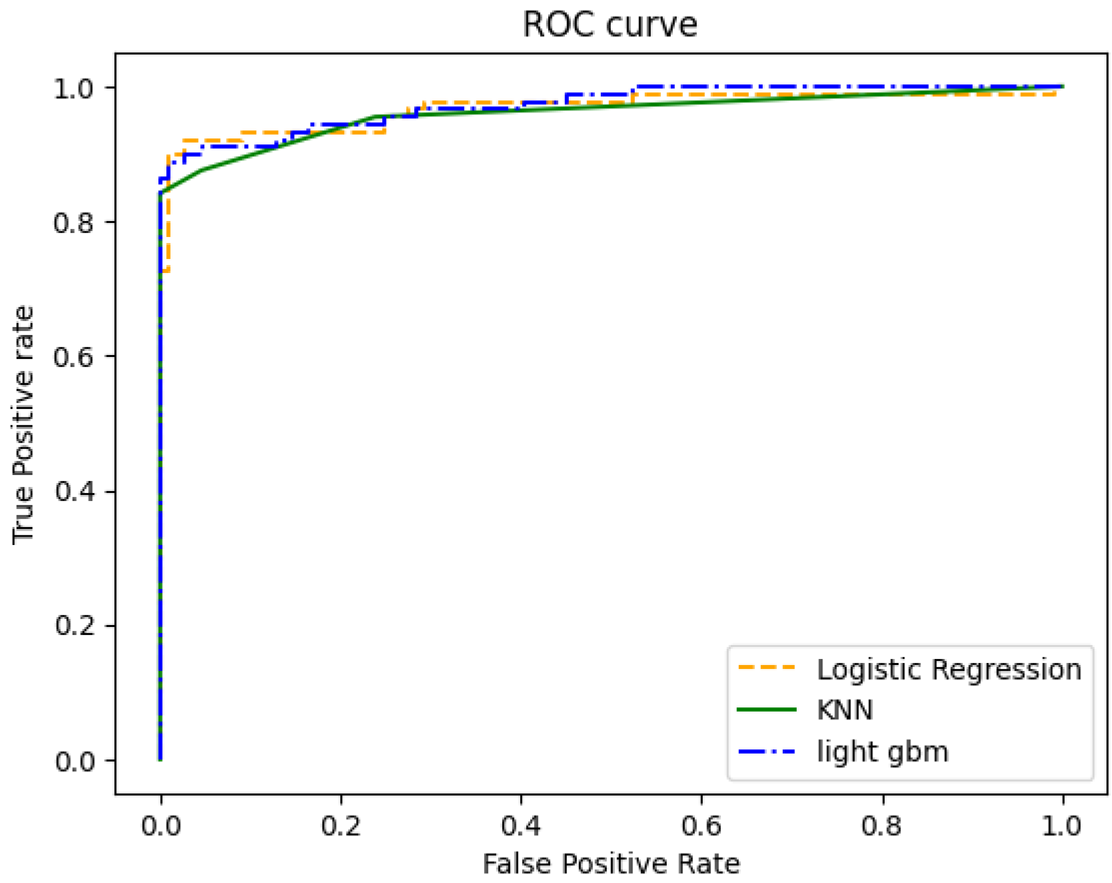
---

	precision	recall	f1-score	support
No Fraud	0.89	1.00	0.94	109
Fraud	1.00	0.84	0.91	88
accuracy			0.93	197



Model performance report
knn_b
Confusion Matrix
1 (Fraud) actual
0 (No Fraud) actual
14
74
109
0
0 (No Fraud) pred
1 (Fraud) pred
Metrics
F1_score
Recall
Precision
Accuracy
0.9136
0.8409
1
0.9289
0
0.5
1
ROC curve (0.96)
true positive rate
false positive rate
Precision - Recall curve
precision
recall
Cumulative gains curve
Percentage positive targeted
Percentage contacted
0.0005
0.001
0.0015
0.002
0.0025
0.003

ROC Curve for lightgbm, Logistic Regression and K-Neighbors Models:



CONCLUSION:

Depending on the specific requirements of problem (e.g., the importance of false positives vs. false negatives), we may choose one model over the other based on the balance between precision and recall. Considering the context of Credit Card Fault Prediction application misclassifications are more costly than others. Hence I am selecting the Approach-3: Resampling and Normalizing all features and lightGBM moedl as this model is giving highest Recall of 0.9091 of among all others and good ROC area of 0.972.

So, I will be using that model to deploy it in AWS sagemaker and integrate the endpoint with AWS API Gateway to publicly access the Invoke endpoint.

2)Model Deployment Process

Docker Image Creation Process for the developed model:

1) Create a folder named “service” containing 3 python scripts (predictor.py, service.py, serve) and nginx.conf file for model deployment

2) Build a docker image of service folder with ‘Dockerfile.sagemaker’ using the following command: docker build . -f Dockerfile.sagemaker -t <image_name>:<tag_name>

   NOTE: In current deployment, <image_name> = faultpredictor and <tag_name> = im

3) Add AWS ECR credentials using following command: aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin <account_id>.dkr.ecr.us-east-1.amazonaws.com

4) If in case docker image_name and tag_name has to be changed use the following command: docker tag <previous_image_name>:<previous_tag_name> <account_id>.dkr.ecr.us-east-1.amazonaws.com/<latest_image_name>:<latest_tag_name>

5) Push the image to AWS ECR using the following command:
   docker push <account_id>.dkr.ecr.us-east-1.amazonaws.com/<latest_image_name>:<latest_tag_name>

   NOTE: In current deployment, <latest_image_name> = faultpredictor and <latest_tag_name> = cc

After Docker Image is pushed to ECR run a “deploy.py”:

6) Now deploy the model in sagemaker and create an endpoint in sagemaker to invoke the model using deploy.py file using the following command:
   python deploy.py --model_data <S3 URI of model location> --image_uri <AWS ECR image uri> --model_server_timeout <timeout> --endpoint <endpoint_name>

   NOTE: In current deployment:

   <S3 URI of model location> = “s3://keysec/cc-experiments/local-developed-model /model/im/model.tar.gz”

   <AWS ECR image uri> = "<account_id>.dkr.ecr.us-east-1.amazonaws.com/faultpredictor:im"

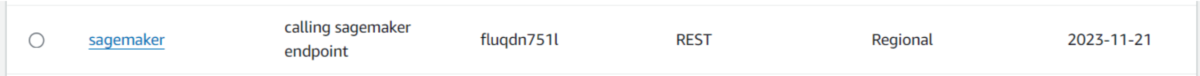
<timeout> = 300

<endpoint_name> = "im"

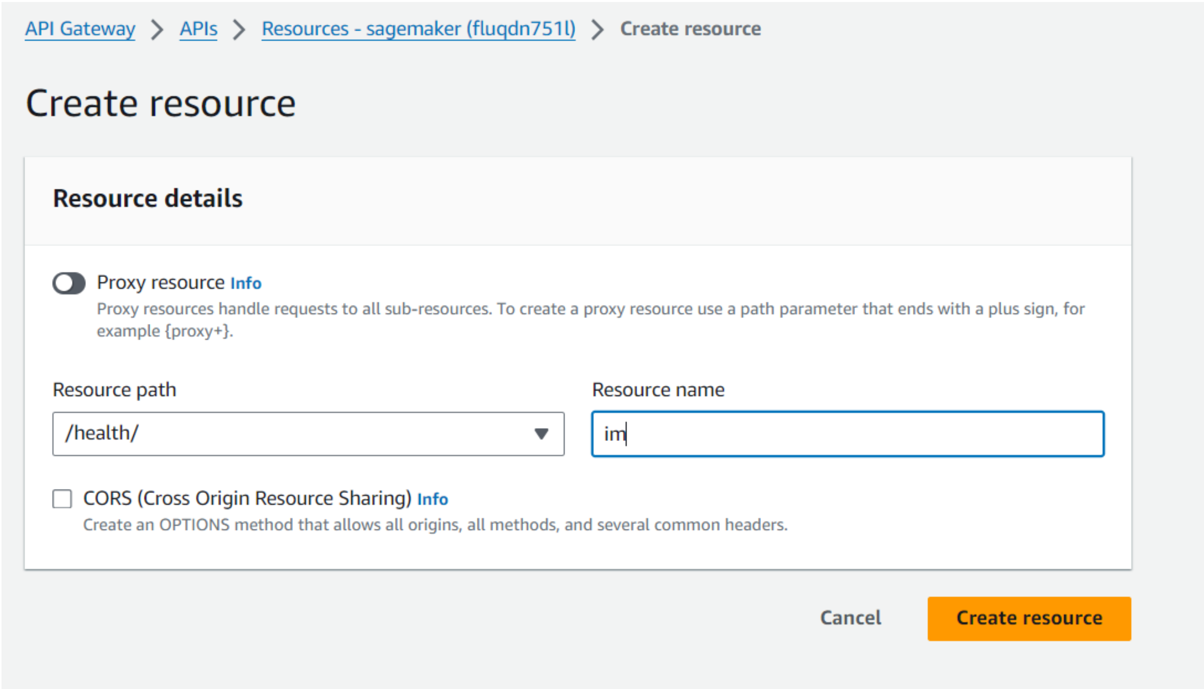
After running "deploy.py" file, Integration of created sagemaker endpoint with API Gateway:

7)

a) In AWS API Gateway select "sagemaker" API

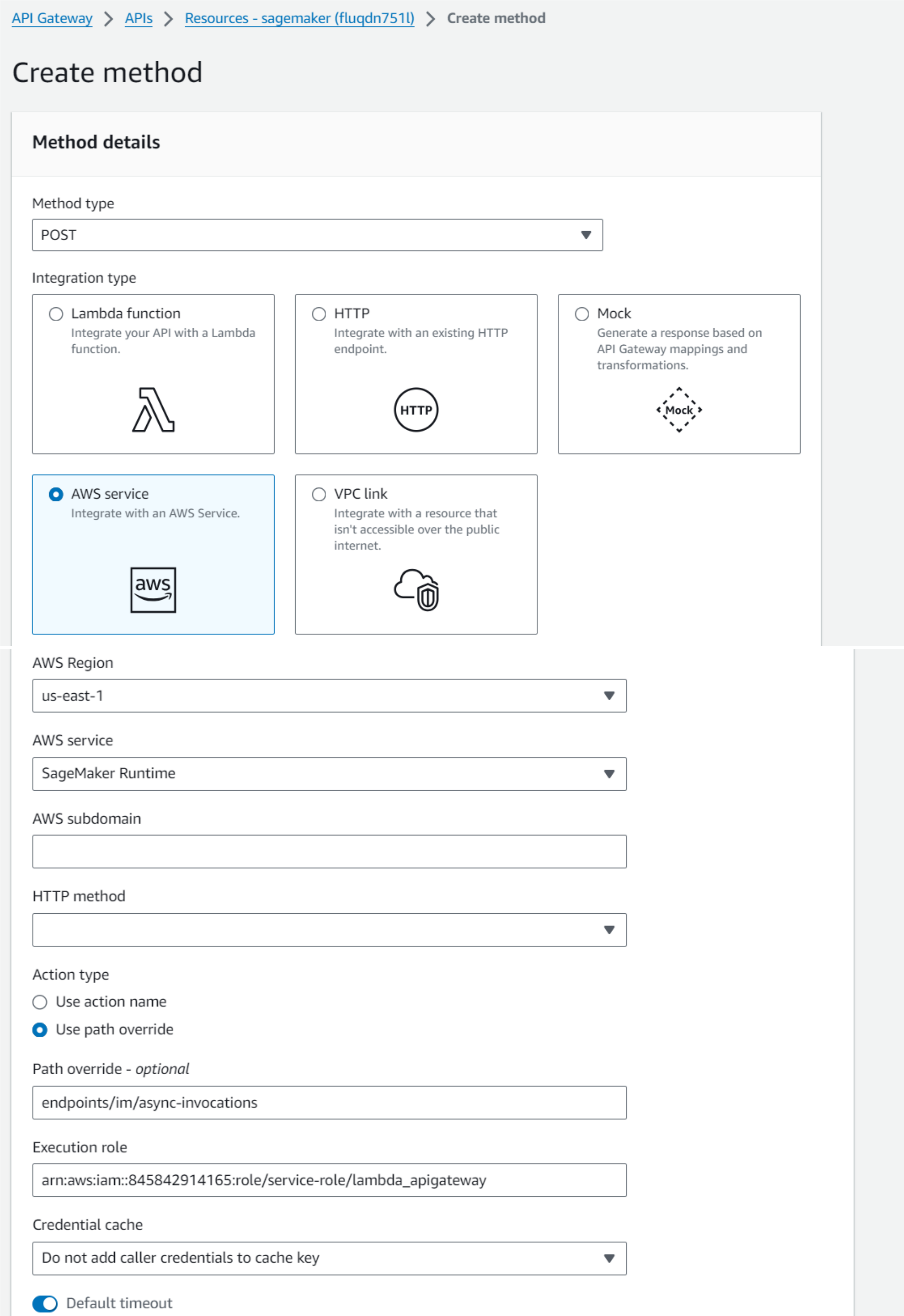


b) In "sagemaker" API, create resource with resource path and resource name



NOTE: In current deployment resource path is "/health/" and resource name is "im"

c) After, create a POST method in that resource with following options :



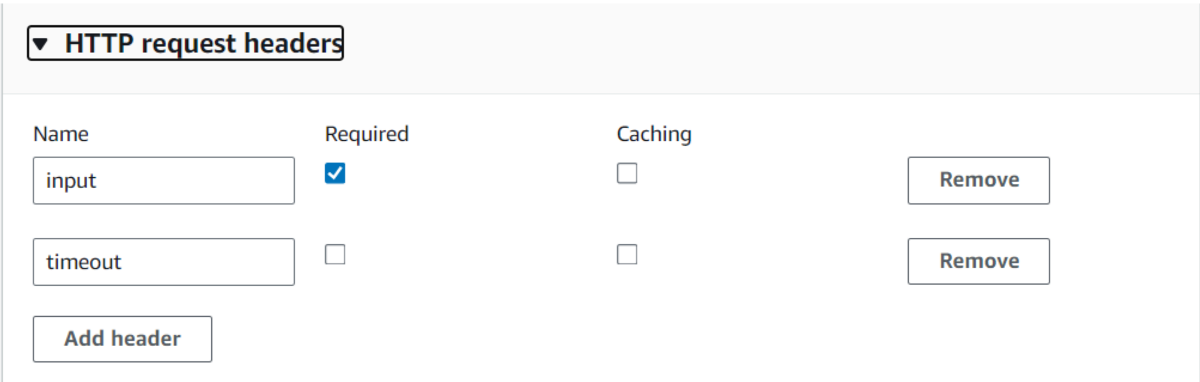
“AWS service” as Integration type,

AWS Region is “us-east-1”,
AWS service is “SageMaker Runtime” ,
select “use path ovveride” in Action type,
in path override use: endpoint/<endpoint_name>/async-invocations,
in Execution role use: arn:aws:iam::<account_id>:role/service-role/lambda_apigateway

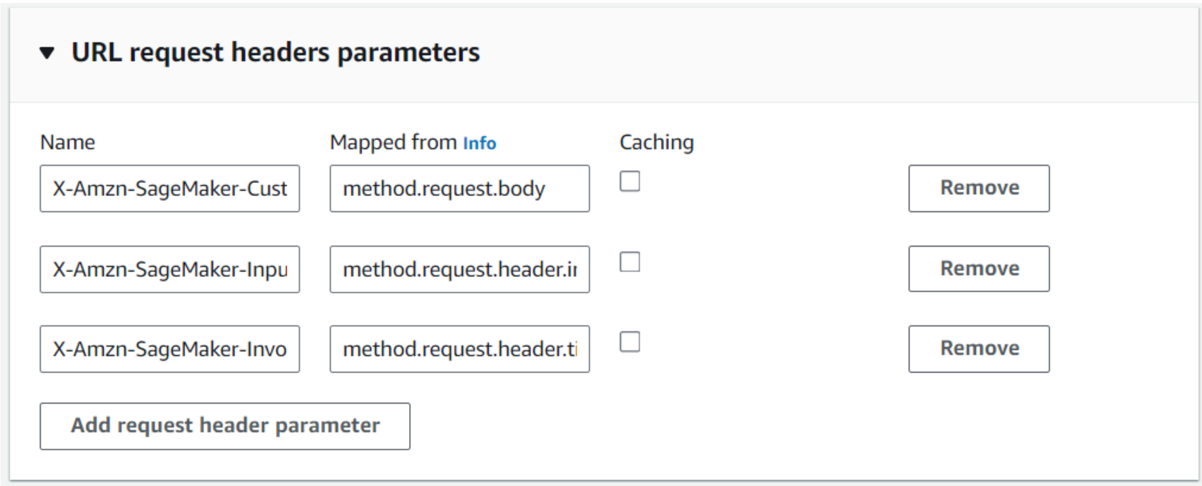
and save it.

NOTE: in current deployment, <endpoint_name> = im, this has to be same as the argument given while running python deploy.py –endpoint <endpoint_name> command previously

d) Edit method request and go to HTTP request headers template add the following headers and save it.

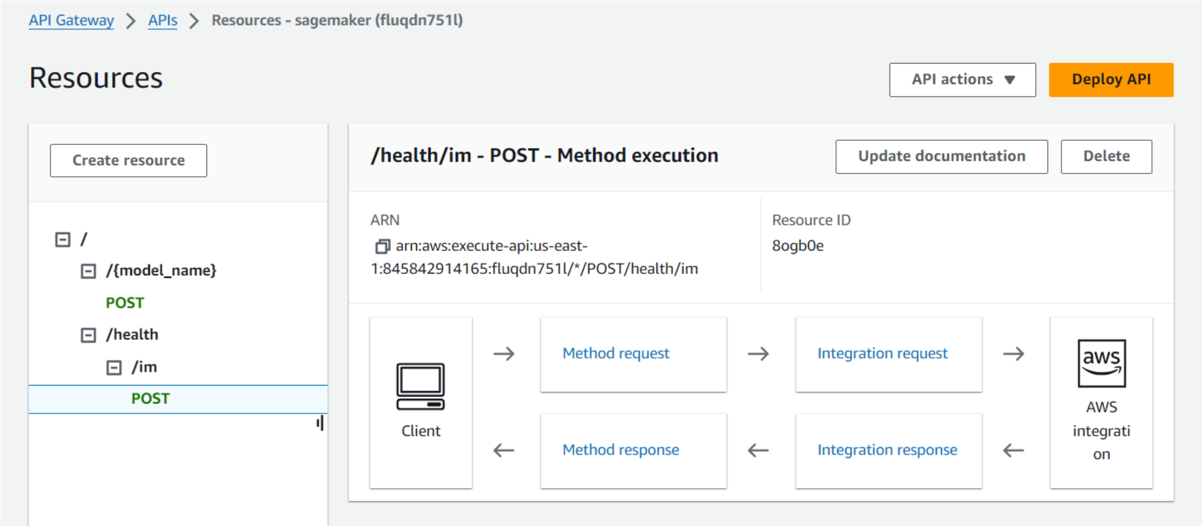


e) Edit Integration request and go to URL request headers parameters template and add the following parameters and save it.

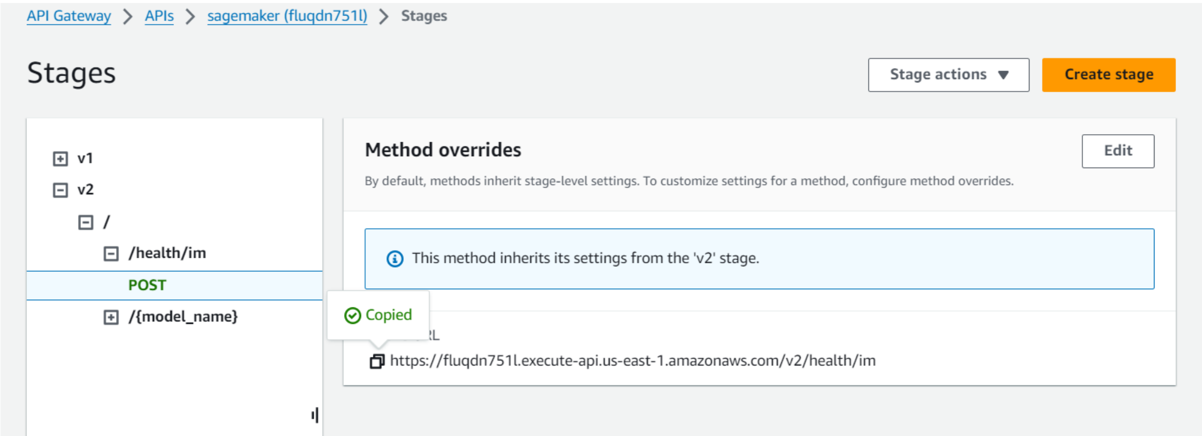


Name	Mapped from
X-Amzn-SageMaker-Custom-Attributes	method.request.body
X-Amzn-SageMaker-InvocationTimeoutSeconds	method.request.header.timeout

f) Deploy the API now.

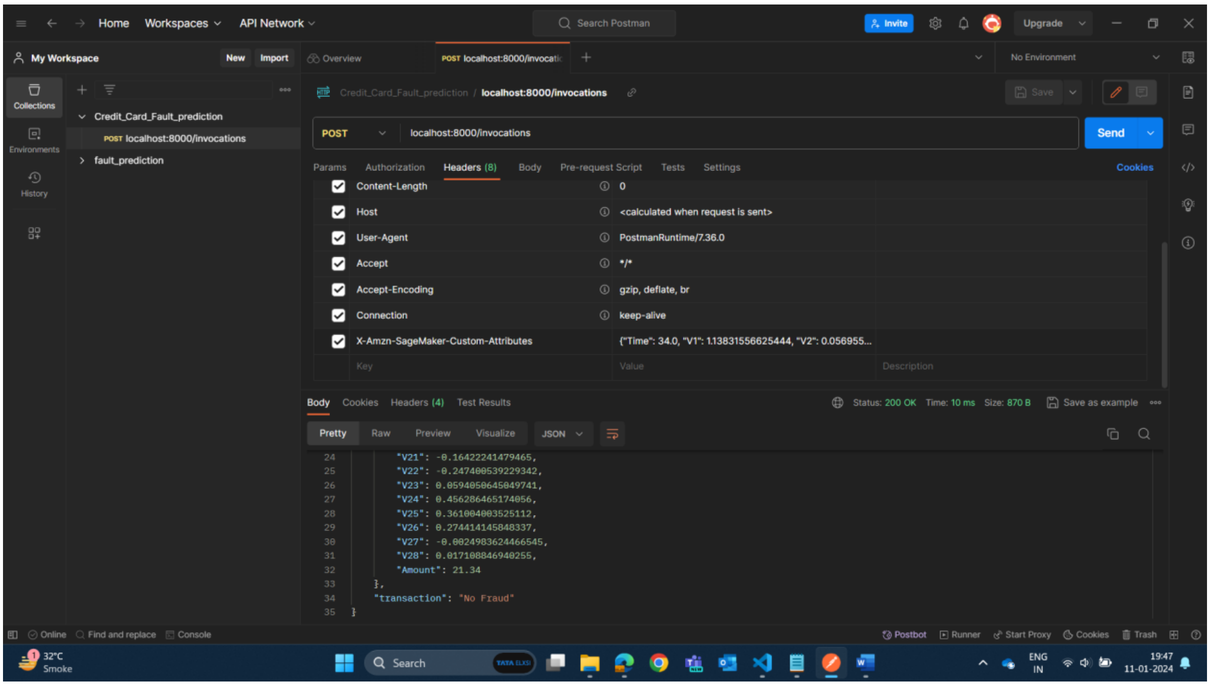


g) Invoke the API by copying the URL

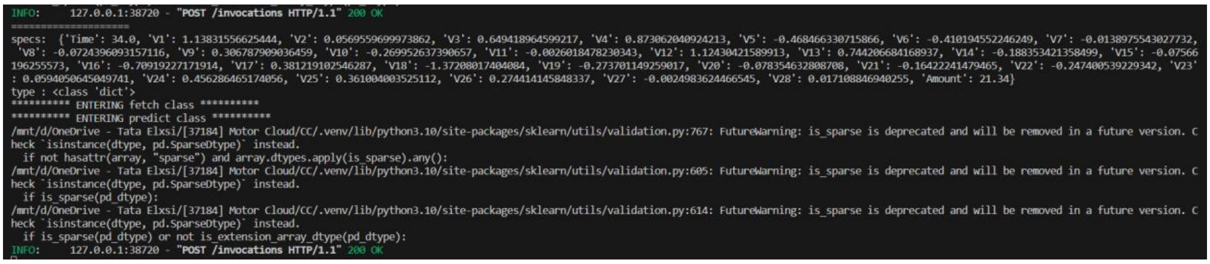


8) Invoke the model using the created URL in API Gateway.

Local Deployment using Fast API and POSTMAN:



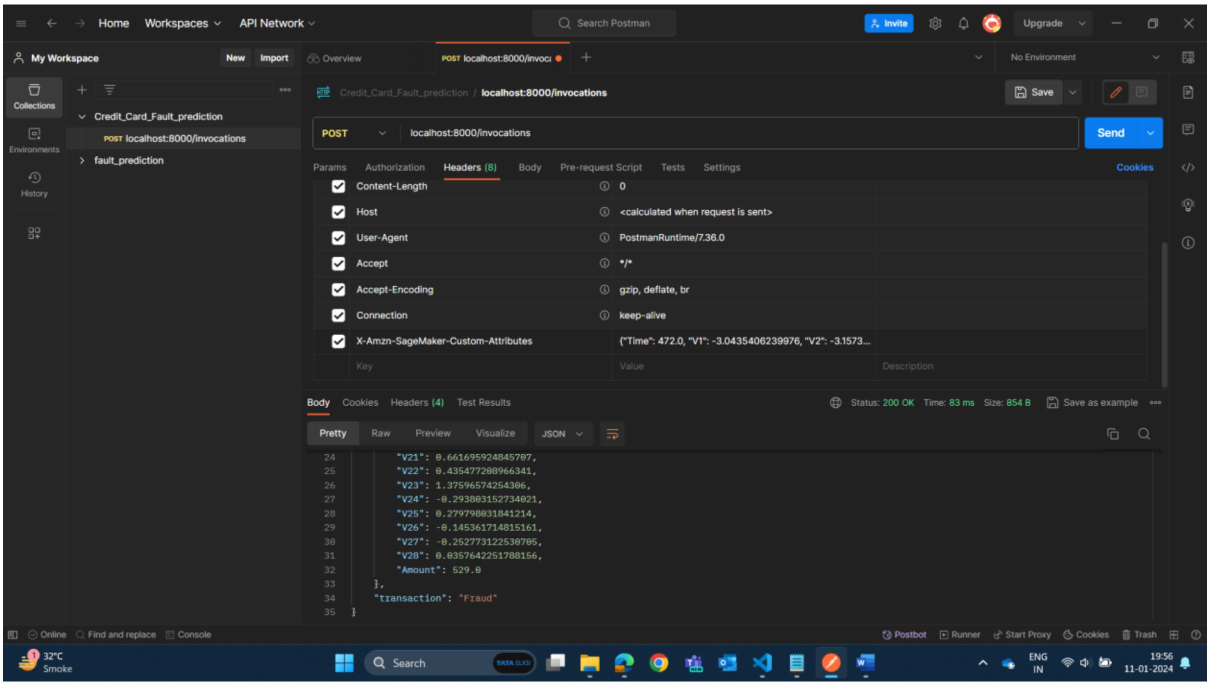
Logs showing the prediction which was invoked above: (Predicted – No Fault as No Fault)



```
INFO:     127.0.0.1:38720 - "POST /invocations HTTP/1.1" 200 OK
====================
specs:  {'Time': 34.0, 'V1': 1.13831556625444, 'V2': 0.0569559699973862, 'V3': 0.649418964599217, 'V4': 0.873062040924213, 'V5': -0.468466330715866, 'V6': -0.410194552246249, 'V7': -0.0138975543027732,
 'V8': -0.0724396093157116, 'V9': 0.306787909036459, 'V10': -0.269952637390657, 'V11': -0.0026018478230343, 'V12': 1.12430421589913, 'V13': 0.744206684168937, 'V14': -0.188353421358499, 'V15': -0.07566
196255573, 'V16': -0.70919227171914, 'V17': 0.381219102546287, 'V18': -1.37208017404084, 'V19': -0.273701149259017, 'V20': -0.078354632808708, 'V21': -0.16422241479465, 'V22': -0.247400539229342, 'V23'
: 0.0594050645049741, 'V24': 0.456286465174056, 'V25': 0.361004003525112, 'V26': 0.274414145848337, 'V27': -0.0024983624466545, 'V28': 0.017108846940255, 'Amount': 21.34}
type : <class 'dict'>
********** ENTERING fetch class **********
********** ENTERING predict class **********
/mnt/d/OneDrive - Tata Elxsi/[37184] Motor Cloud/CC/.venv/lib/python3.10/site-packages/sklearn/utils/validation.py:767: FutureWarning: is_sparse is deprecated and will be removed in a future version. C
heck `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/mnt/d/OneDrive - Tata Elxsi/[37184] Motor Cloud/CC/.venv/lib/python3.10/site-packages/sklearn/utils/validation.py:605: FutureWarning: is_sparse is deprecated and will be removed in a future version. C
heck `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/mnt/d/OneDrive - Tata Elxsi/[37184] Motor Cloud/CC/.venv/lib/python3.10/site-packages/sklearn/utils/validation.py:614: FutureWarning: is_sparse is deprecated and will be removed in a future version. C
heck `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
INFO:     127.0.0.1:38720 - "POST /invocations HTTP/1.1" 200 OK
```

RESPONSE BODY of above request:


```json
{
    "features": {
        "Time": 34.0,
        "V1": 1.13831556625444,
        "V2": 0.0569559699973862,
        "V3": 0.649418964599217,
        "V4": 0.873062040924213,
        "V5": -0.468466330715866,
        "V6": -0.410194552246249,
        "V7": -0.0138975543027732,
        "V8": -0.0724396093157116,
        "V9": 0.306787909036459,
        "V10": -0.269952637390657,
        "V11": -0.0026018478230343,
        "V12": 1.12430421589913,
        "V13": 0.744206684168937,
        "V14": -0.188353421358499,
        "V15": -0.07566196255573,
        "V16": -0.70919227171914,
        "V17": 0.381219102546287,
        "V18": -1.37208017404084,
        "V19": -0.273701149259017,
        "V20": -0.078354632808708,
        "V21": -0.16422241479465,
        "V22": -0.247400539229342,
        "V23": 0.0594050645049741,
        "V24": 0.456286465174056,
        "V25": 0.361004003525112,
        "V26": 0.274414145848337,
        "V27": -0.0024983624466545,
        "V28": 0.017108846940255,
        "Amount": 21.34
    },
    "transaction": "No Fraud"
}
```



Logs showing the prediction which was invoked above: (Predicted – Fault as Fault)

```
INFO:     127.0.0.1:38720 - "POST /invocations HTTP/1.1" 200 OK
====================
specs:  {'Time': 472.0, 'V1': -3.0435406239976, 'V2': -3.15730712090228, 'V3': 1.08846277997285, 'V4': 2.2886436183814, 'V5': 1.35980512966107, 'V6': -1.06482252298131, 'V7': 0.325574266158614, 'V8': -
0.0677936531906277, 'V9': -0.270952836226548, 'V10': -0.838586564582682, 'V11': -0.414575448285725, 'V12': -0.503140859566824, 'V13': 0.676501544635863, 'V14': -1.69202893305906, 'V15': 2.0006348390901
5, 'V16': 0.666779695901966, 'V17': 0.599717413841732, 'V18': 1.72532100745514, 'V19': 0.283344830149495, 'V20': 2.10233879259444, 'V21': 0.661695924845707, 'V22': 0.435477208966341, 'V23': 1.375965742
54306, 'V24': -0.293803152734021, 'V25': 0.279798031841214, 'V26': -0.145361714815161, 'V27': -0.252773122530705, 'V28': 0.0357642251788156, 'Amount': 529.0}
type : <class 'dict'>
********** ENTERING fetch class **********
********** ENTERING predict class **********
/mnt/d/OneDrive - Tata Elxsi/[37184] Motor Cloud/CC/.venv/lib/python3.10/site-packages/sklearn/utils/validation.py:767: FutureWarning: is_sparse is deprecated and will be removed in a future version. C
heck `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/mnt/d/OneDrive - Tata Elxsi/[37184] Motor Cloud/CC/.venv/lib/python3.10/site-packages/sklearn/utils/validation.py:605: FutureWarning: is_sparse is deprecated and will be removed in a future version. C
heck `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/mnt/d/OneDrive - Tata Elxsi/[37184] Motor Cloud/CC/.venv/lib/python3.10/site-packages/sklearn/utils/validation.py:614: FutureWarning: is_sparse is deprecated and will be removed in a future version. C
heck `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
INFO:     127.0.0.1:40644 - "POST /invocations HTTP/1.1" 200 OK
```

RESPONSE BODY of above request:


```json
{
    "features": {
        "Time": 472.0,
        "V1": -3.0435406239976,
        "V2": -3.15730712090228,
        "V3": 1.08846277997285,
        "V4": 2.2886436183814,
        "V5": 1.35980512966107,
        "V6": -1.06482252298131,
        "V7": 0.325574266158614,
        "V8": -0.0677936531906277,
        "V9": -0.270952836226548,
        "V10": -0.838586564582682,
        "V11": -0.414575448285725,
        "V12": -0.503140859566824,
        "V13": 0.676501544635863,
        "V14": -1.69202893305906,
        "V15": 2.00063483909015,
        "V16": 0.666779695901966,
        "V17": 0.599717413841732,
        "V18": 1.72532100745514,
        "V19": 0.283344830149495,
        "V20": 2.10233879259444,
        "V21": 0.661695924845707,
        "V22": 0.435477208966341,
        "V23": 1.37596574254306,
        "V24": -0.293803152734021,
        "V25": 0.279798031841214,
        "V26": -0.145361714815161,
        "V27": -0.252773122530705,
        "V28": 0.0357642251788156,
        "Amount": 529.0
    },
    "transaction": "Fraud"
}
```

Future Roadmap

3.1) Ensemble model prediction

To go through the diverse algorithms, an ensemble approach is adopted which will include models like Adaboost, XGboost, Catboost, LGBM. Combining diverse models can capture different patterns in the data. And later, stacking will be done, stacking often leads to better predictive performance compared to individual models.

3.2) Aggregation of Analytical and Machine Learning Approach

Implement a voting mechanism to aggregate predictions.

3.2.1) Voting based aggregation:

Predicted class is the one that receives most votes from the models.

3.2.2) Weightage based aggregation:

Assign different weights of the predictions of each model based on their performance or reliability.