Project Report

On

Analysis and Prediction of Credit Loan Default

Submitted in partial fulfilment for the award of

PG-Diploma in Big Data Analytics

from **C-DAC ACTS (Pune)**

Guided by:

Mr. Prateek Maheshwari

Presented by:

Mr. Balaji Shewale	**PRN: 210940125011**
Mr. Harsh Mohan	**PRN: 210940125019**
Mr. Balkrishna Nandavadekar	**PRN: 210940125027**
Mr. Omkar Lokhande	**PRN: 210940125030**
Mr. Vedant Deshpande	**PRN: 210940125053**

Centre of Development of Advanced Computing (C-DAC), Pune

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Mr. Balaji Shewale

Mr. Harsh Mohan

Mr. Balkrishna Nandavadekar

Mr. Omkar Lokhande

Mr. Vedant Deshpande

have successfully completed their project on

Analysis and Prediction of Credit Loan Default

under the guidance of Mr. Prateek Maheshwari

Program Head	Project Supervisor
Ms Risha P.R.	Mr. Prateek Maheshwari

HOD ACTS

Mr. Sundar Gaur

ACKNOWLEDGEMENT

This project “Analysis and Prediction of Credit Loan Default” was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS). We all are very glad to mention the name of Mr. Prateek Maheshwari for his valuable guidance to work on this project. His guidance and support helped us to overcome various obstacles and intricacies during the course of project work.

We are highly grateful to Ms. Risha P.R. (Manager (ACTS training Centre), C-DAC), for her guidance and support whenever necessary while doing this course PG-Diploma in Big Data Analytics (PG-DBDA) through C-DAC ACTS, Pune.

Our most heartfelt thanks go to Ms. Seema Sajeevan (Course Coordinator, PG- DBDA) who gave all the required support and kind coordination to provide all the necessities where needed.

1.Abstract

Credit risk plays a major role in the banking industry business. Credit card has been one of the most booming financial services by banks over the past years. However, with the growing number of credit card users, banks have been facing an escalating credit card default rate.

This study aims at using data of the customer to predict whether the customer will default by various statistical and data mining techniques and building different models for the same. The exploratory data analysis part is also important to check the distributions and patterns followed by the customers which eventually lead to default. We have experimented with various machine learning algorithms using mlflow for experiment tracking and Artificial Neural Network provided the best results. The model was deployed using Flask

2. Introduction

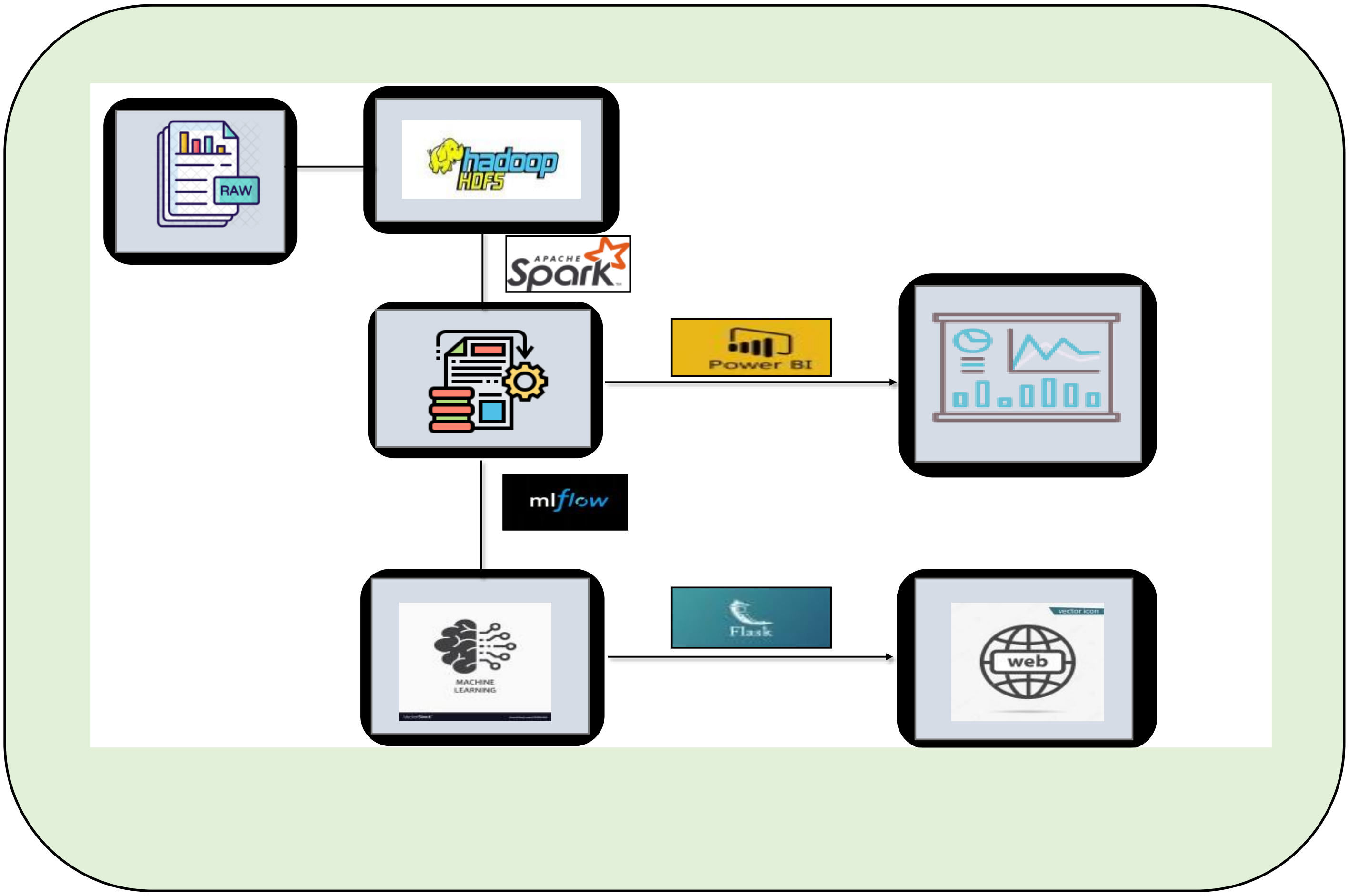
When a customer applies for and receive a credit card, it becomes a huge responsibility for customer as well as the credit card issuing company. The credit card company evaluates the customer's credit worthiness and gives him/her a line of credit that they feel the customer can be responsible for. While most people will use their card to make purchases and then diligently make payments on what they charge, there are some people who, for one reason or another, do not keep up on their payments and eventually go into credit card default.

Credit card default is the term used to describe what happens when a credit card user makes purchases by charging them to their credit card and then they do not pay their bill. It can occur when one payment is more than 30 days past due, which may raise your interest rate. Most of the time, the term default is used informally when the credit card payment is more than 60 days past due. A default has a negative impact on the credit report and most likely lead to higher interest rates on future borrowing.

In recent years, the credit card issuers are facing the cash and credit card debt crisis as they have been over-issuing cash and credit cards to unqualified applicants, in order to increase their market share. At the same time, most cardholders, irrespective of their repayment ability, overused credit card for consumption and accumulated heavy credit and cash-card debts. The crisis is an omen for the blow to consumer finance confidence and it is a big challenge for both banks and cardholders.

3. Flowchart

.


RAW
hadoop HDFS
APACHE Spark
Power BI
mlflow
MACHINE LEARNING
Flask
web

4. Information on Technologies used

Python:

Python is a high-level, general-purpose computer programming language. It is often used to build websites and software, automate tasks, and for data cleaning & analysis. For this project, we have used Python to cleaned the data, for plotting graphs and charts and to for Machine learning. We will be using the libraries pandas, NumPy, Spark, Mlflow, Scikit etc. The rest already come with the Python interpreter.

HDFS:

HDFS (Hadoop Distributed File System) is a distributed file system that handles large data sets running on commodity hardware. HDFS has been built to detect faults and automatically recover quickly. HDFS is used more for batch processing and it is built on write-once and read-many-times pattern. We have used HDFS to store the Raw (Uncleaned) data and to integrate it with Apache Spark.

Apache Spark:

Apache Spark is an open-source, distributed processing system which utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size. Spark can be deployed in a variety of ways, provides native bindings for the Java, Scala, Python, and R programming languages, and supports SQL, streaming data, machine learning, and graph processing. We have used Apache Spark to load data from HDFS and for Data Cleaning Process.

Power BI:

Power BI is Microsoft's interactive data visualization and analytics tool for business intelligence (BI). Power BI Desktop lets you connect to, transform, and visualize your data and helps easily to gain actionable insights. We have used Powe Bi to build a Dashboard to Analyse and to discover insights from the cleaned dataset

Mlflow:

Mlflow is an open-source platform for managing the end-to-end machine learning lifecycle. It provides a powerful way to simplify and linearly expand the deployment of machine learning by tracking, reproducing, managing and deploying models in software development

Flask:

Flask is a a web application framework written in python, in simple terms it helps to deploy the model directly from their web browser without needing any libraries, code files, etc. The main purpose of this application is to determine if an individual is a defaulter or not and display it on a webserver

Amazon S3 and EC2 instance:

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. We have used it to store the Cleaned data and then to integrate with Power BI to create Dashboard and gain insights. AWS EC2 is a virtual server we have deployed our web application.

5. Dataset Description

We have imported the Credit card default dataset from KAGGLE.

Content:

It contains the following 19 fields:

1. Customer id
2. name
3. age
4. gender
5. owns car
6. owns_house
7. no_of_children
8. net_yearly_income
9. no_of_days_employed
10. occupation_type
11. total_family_members
12. migrant_worker

13. yearly_debt_payments

14. credit_limit

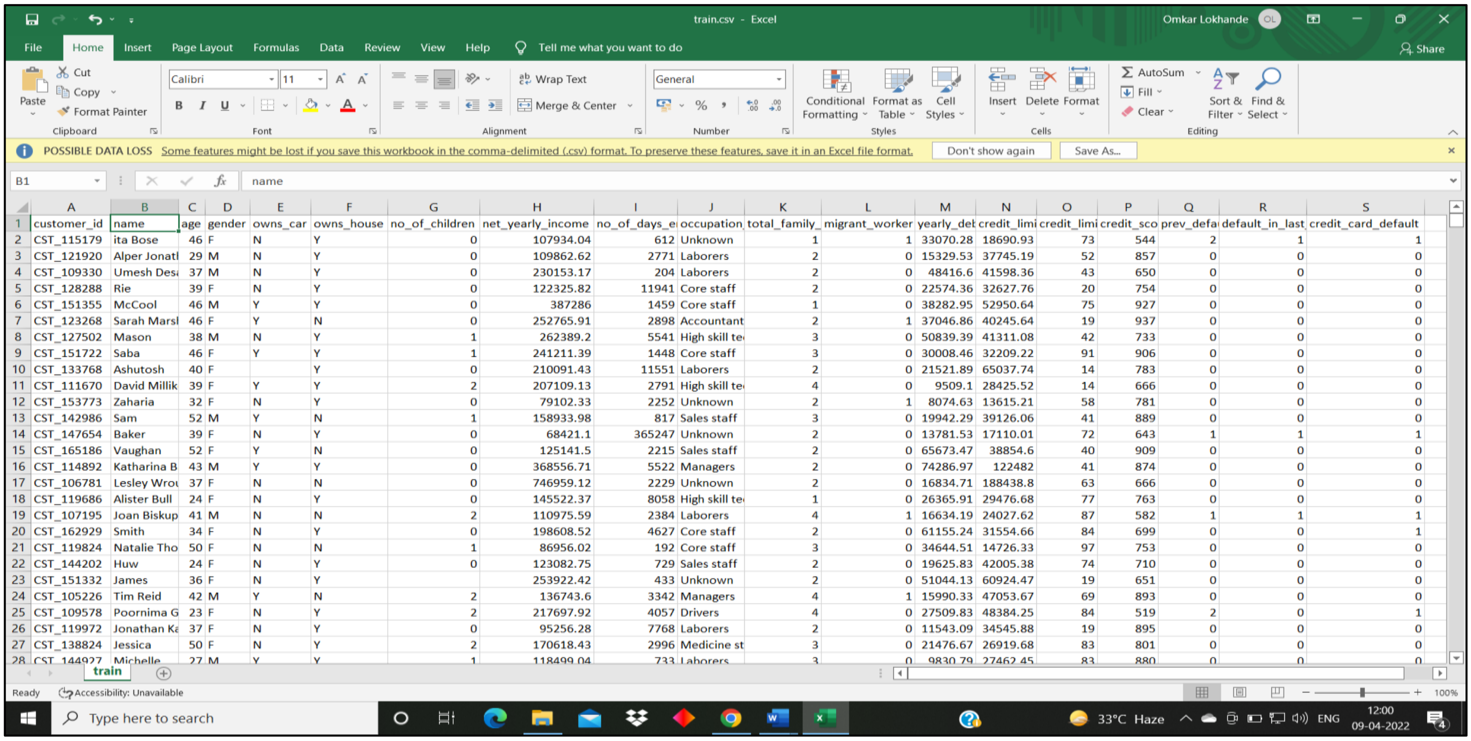
15. credit_limit_used(%)

16. credit_score

17. prev_defaults

18. default_in_last_6months

19. credit_card_default



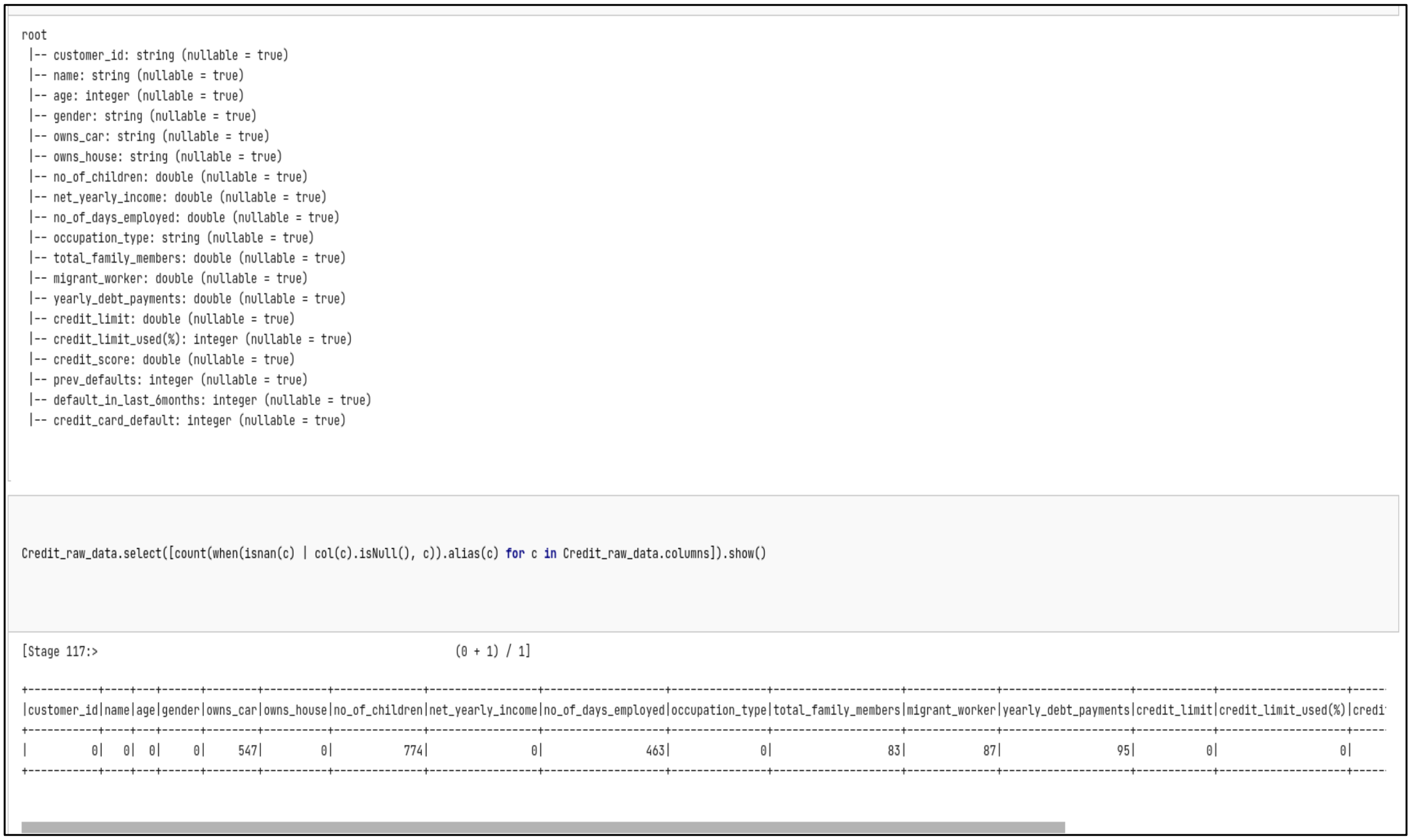
POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

B1 name

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	customer_id	name	age	gender	owns_car	owns_house	no_of_children	net_yearly_income	no_of_days_e	occupation_	total_family_	migrant_worker	yearly_det	credit_limi	credit_limi	credit_sco	prev_defa	default_in_last	credit_card_default
2	CST_115179	ita Bose	46	F	N	Y	0	107934.04	612	Unknown	1	1	33070.28	18690.93	73	544	2	1	1
3	CST_121920	Alper Jonatl	29	M	N	Y	0	109862.62	2771	Laborers	2	0	15329.53	37745.19	52	857	0	0	0
4	CST_109330	Umesh Desa	37	M	N	Y	0	230153.17	204	Laborers	2	0	48416.6	41598.36	43	650	0	0	0
5	CST_128288	Rie	39	F	N	Y	0	122325.82	11941	Core staff	2	0	22574.36	32627.76	20	754	0	0	0
6	CST_151355	McCool	46	M	Y	Y	0	387286	1459	Core staff	1	0	38282.95	52950.64	75	927	0	0	0
7	CST_123268	Sarah Marsl	46	F	Y	N	0	252765.91	2898	Accountant	2	1	37046.86	40245.64	19	937	0	0	0
8	CST_127502	Mason	38	M	N	Y	1	262389.2	5541	High skill te	3	0	50839.39	41311.08	42	733	0	0	0
9	CST_151722	Saba	46	F	Y	Y	1	241211.39	1448	Core staff	3	0	30008.46	32209.22	91	906	0	0	0
10	CST_133768	Ashutosh	40	F		Y	0	210091.43	11551	Laborers	2	0	21521.89	65037.74	14	783	0	0	0
11	CST_111670	David Millik	39	F	Y	Y	2	207109.13	2791	High skill te	4	0	9509.1	28425.52	14	666	0	0	0
12	CST_153773	Zaharia	32	F	N	Y	0	79102.33	2252	Unknown	2	1	8074.63	13615.21	58	781	0	0	0
13	CST_142986	Sam	52	M	Y	N	1	158933.98	817	Sales staff	3	0	19942.29	39126.06	41	889	0	0	0
14	CST_147654	Baker	39	F	N	Y	0	68421.1	365247	Unknown	2	0	13781.53	17110.01	72	643	1	1	1
15	CST_165186	Vaughan	52	F	Y	N	0	125141.5	2215	Sales staff	2	0	65673.47	38854.6	40	909	0	0	0
16	CST_114892	Katharina B	43	M	Y	Y	0	368556.71	5522	Managers	2	0	74286.97	122482	41	874	0	0	0
17	CST_106781	Lesley Wrou	37	F	N	N	0	746959.12	2229	Unknown	2	0	16834.71	188438.8	63	666	0	0	0
18	CST_119686	Alister Bull	24	F	N	Y	0	145522.37	8058	High skill te	1	0	26365.91	29476.68	77	763	0	0	0
19	CST_107195	Joan Biskup	41	M	N	N	2	110975.59	2384	Laborers	4	1	16634.19	24027.62	87	582	1	1	1
20	CST_162929	Smith	34	F	N	Y	0	198608.52	4627	Core staff	2	0	61155.24	31554.66	84	699	0	0	1
21	CST_119824	Natalie Tho	50	F	N	N	1	86956.02	192	Core staff	3	0	34644.51	14726.33	97	753	0	0	0
22	CST_144202	Huw	24	F	N	Y	0	123082.75	729	Sales staff	2	0	19625.83	42005.38	74	710	0	0	0
23	CST_151332	James	36	F	N	Y		253922.42	433	Unknown	2	0	51044.13	60924.47	19	651	0	0	0
24	CST_105226	Tim Reid	42	M	Y	N	2	136743.6	3342	Managers	4	1	15990.33	47053.67	69	893	0	0	0
25	CST_109578	Poornima G	23	F	N	Y	2	217697.92	4057	Drivers	4	0	27509.83	48384.25	84	519	2	0	1
26	CST_119972	Jonathan Ka	37	F	N	Y	0	95256.28	7768	Laborers	2	0	11543.09	34545.88	19	895	0	0	0
27	CST_138824	Jessica	50	F	N	Y	2	170618.43	2996	Medicine st	3	0	21476.67	26919.68	83	801	0	0	0
28	CST_144927	Michelle	27	M	Y	Y	1	118499.04	733	Laborers	3	0	9830.79	27462.45	83	880	0	0	0

6. Data Pre-processing

Following is the snippet of the column names and their respective datatypes along with the count of NAN and Null values present in each column.

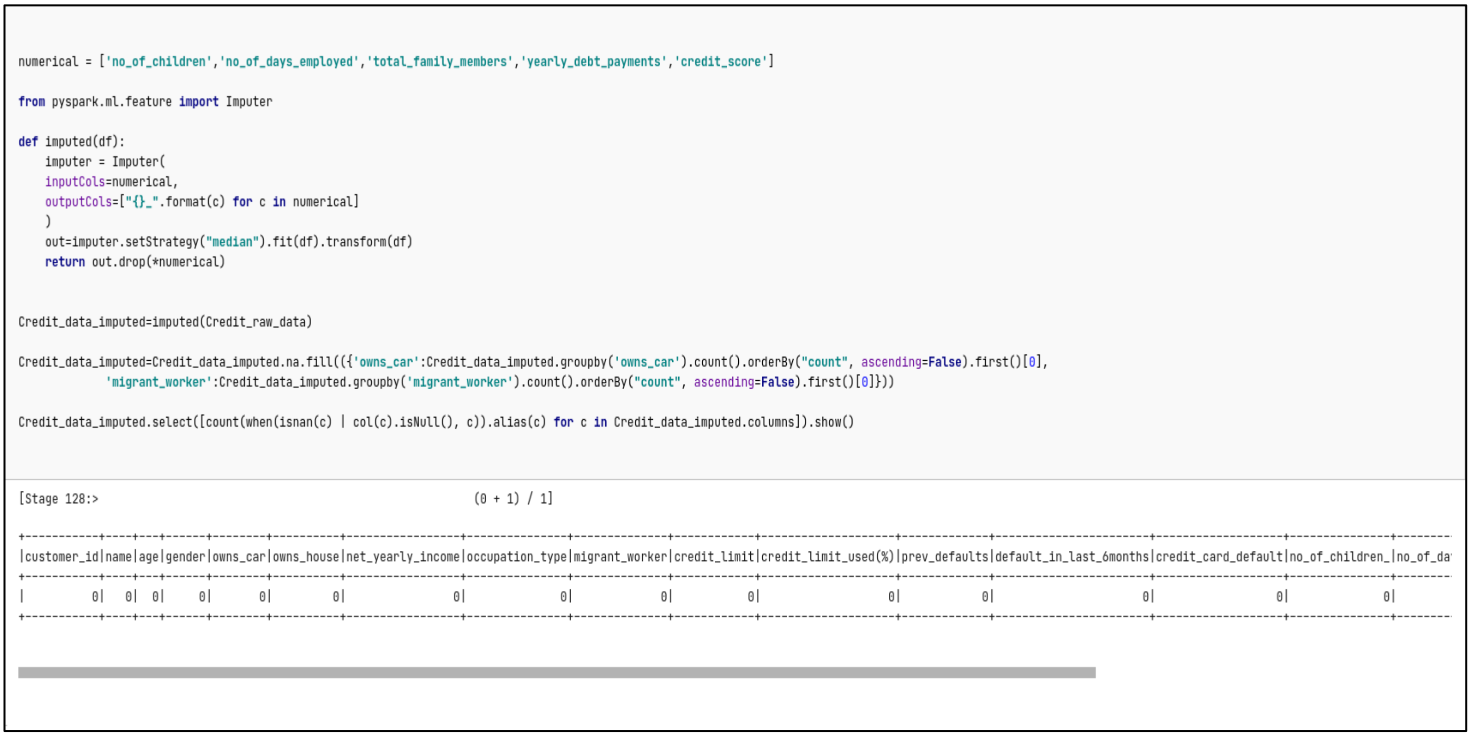


```
root
 |-- customer_id: string (nullable = true)
 |-- name: string (nullable = true)
 |-- age: integer (nullable = true)
 |-- gender: string (nullable = true)
 |-- owns_car: string (nullable = true)
 |-- owns_house: string (nullable = true)
 |-- no_of_children: double (nullable = true)
 |-- net_yearly_income: double (nullable = true)
 |-- no_of_days_employed: double (nullable = true)
 |-- occupation_type: string (nullable = true)
 |-- total_family_members: double (nullable = true)
 |-- migrant_worker: double (nullable = true)
 |-- yearly_debt_payments: double (nullable = true)
 |-- credit_limit: double (nullable = true)
 |-- credit_limit_used(%): integer (nullable = true)
 |-- credit_score: double (nullable = true)
 |-- prev_defaults: integer (nullable = true)
 |-- default_in_last_6months: integer (nullable = true)
 |-- credit_card_default: integer (nullable = true)
```

```
Credit_raw_data.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in Credit_raw_data.columns]).show()
```

```
[Stage 117:>                                                        (0 + 1) / 1]

+-----------+----+---+------+--------+----------+--------------+-----------------+-------------------+---------------+--------------------+--------------+--------------------+------------+--------------------+-----
|customer_id|name|age|gender|owns_car|owns_house|no_of_children|net_yearly_income|no_of_days_employed|occupation_type|total_family_members|migrant_worker|yearly_debt_payments|credit_limit|credit_limit_used(%)|credi
+-----------+----+---+------+--------+----------+--------------+-----------------+-------------------+---------------+--------------------+--------------+--------------------+------------+--------------------+-----
|          0|   0|  0|     0|     547|         0|           774|                0|                463|              0|                  83|            87|                  95|           0|                   0|
+-----------+----+---+------+--------+----------+--------------+-----------------+-------------------+---------------+--------------------+--------------+--------------------+------------+--------------------+-----
```

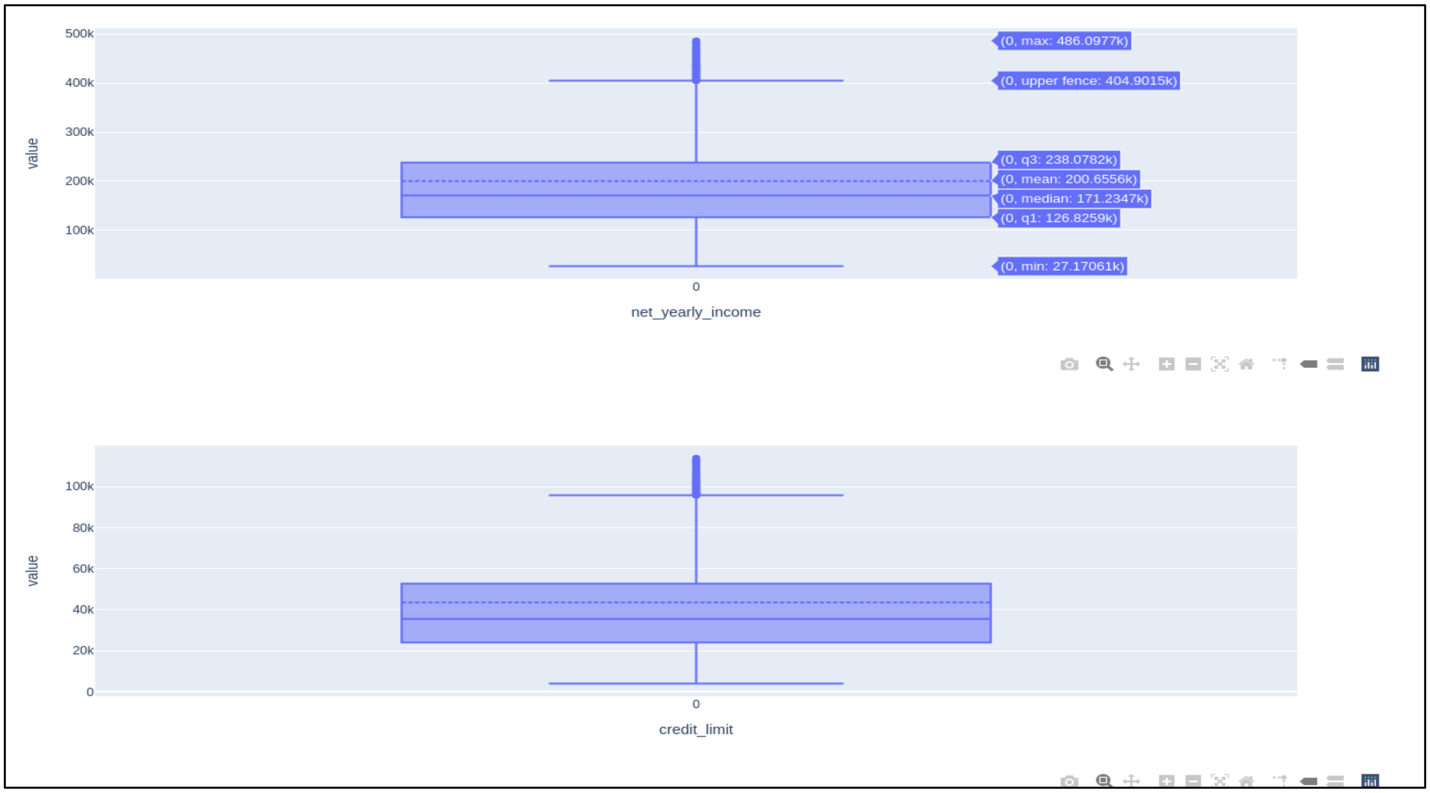
Imputing the NAN and Null values with the median



```
numerical = ['no_of_children','no_of_days_employed','total_family_members','yearly_debt_payments','credit_score']

from pyspark.ml.feature import Imputer

def imputed(df):
    imputer = Imputer(
    inputCols=numerical,
    outputCols=["{}_".format(c) for c in numerical]
    )
    out=imputer.setStrategy("median").fit(df).transform(df)
    return out.drop(*numerical)


Credit_data_imputed=imputed(Credit_raw_data)

Credit_data_imputed=Credit_data_imputed.na.fill(({'owns_car':Credit_data_imputed.groupby('owns_car').count().orderBy("count", ascending=False).first()[0],
             'migrant_worker':Credit_data_imputed.groupby('migrant_worker').count().orderBy("count", ascending=False).first()[0]}))

Credit_data_imputed.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in Credit_data_imputed.columns]).show()
```

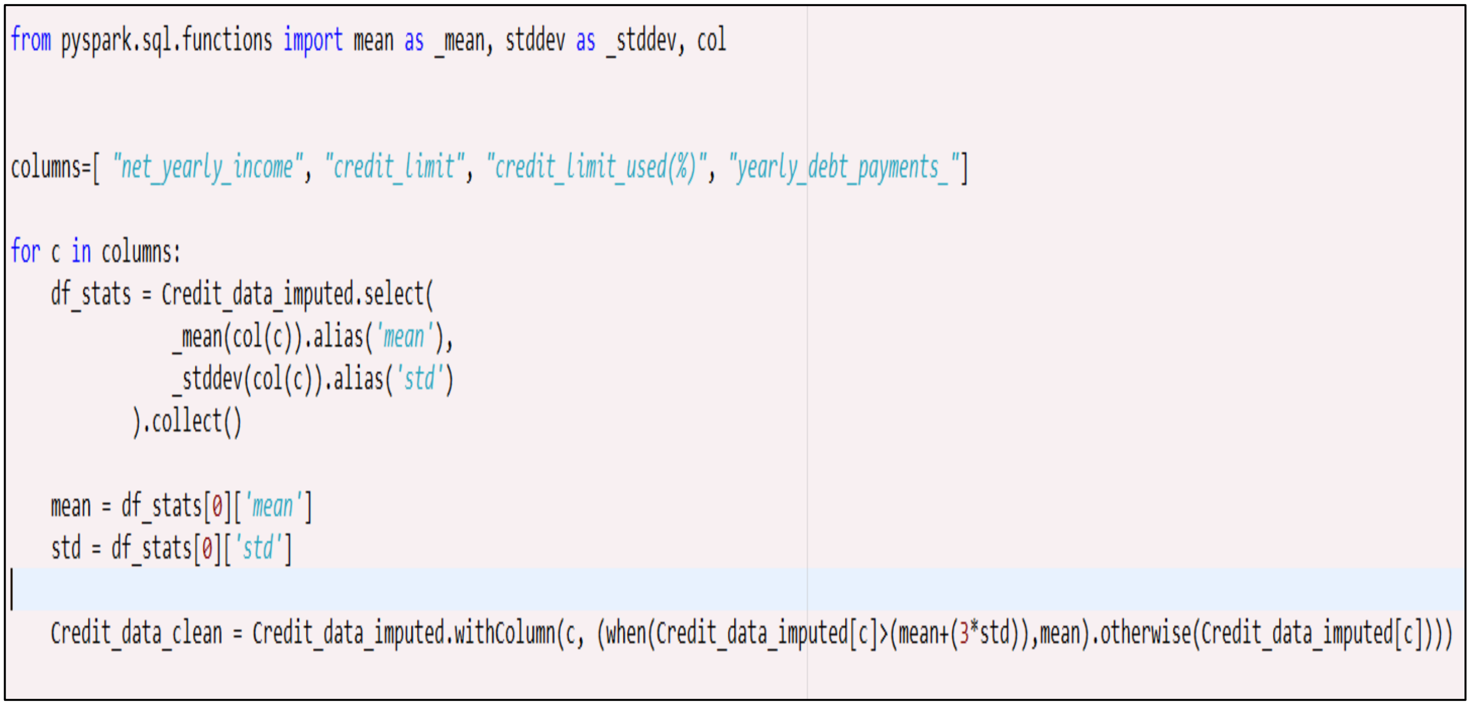
```
[Stage 128:>                                                        (0 + 1) / 1]

+-----------+----+---+------+--------+----------+-----------------+---------------+--------------+------------+--------------------+-------------+-----------------------+-------------------+---------------+--------
|customer_id|name|age|gender|owns_car|owns_house|net_yearly_income|occupation_type|migrant_worker|credit_limit|credit_limit_used(%)|prev_defaults|default_in_last_6months|credit_card_default|no_of_children_|no_of_da
+-----------+----+---+------+--------+----------+-----------------+---------------+--------------+------------+--------------------+-------------+-----------------------+-------------------+---------------+--------
|          0|   0|  0|     0|       0|         0|                0|              0|             0|           0|                   0|            0|                      0|                  0|              0|
+-----------+----+---+------+--------+----------+-----------------+---------------+--------------+------------+--------------------+-------------+-----------------------+-------------------+---------------+--------
```

Using Box-Plot to detect the outliers in various columns



Replacing the outlier values:

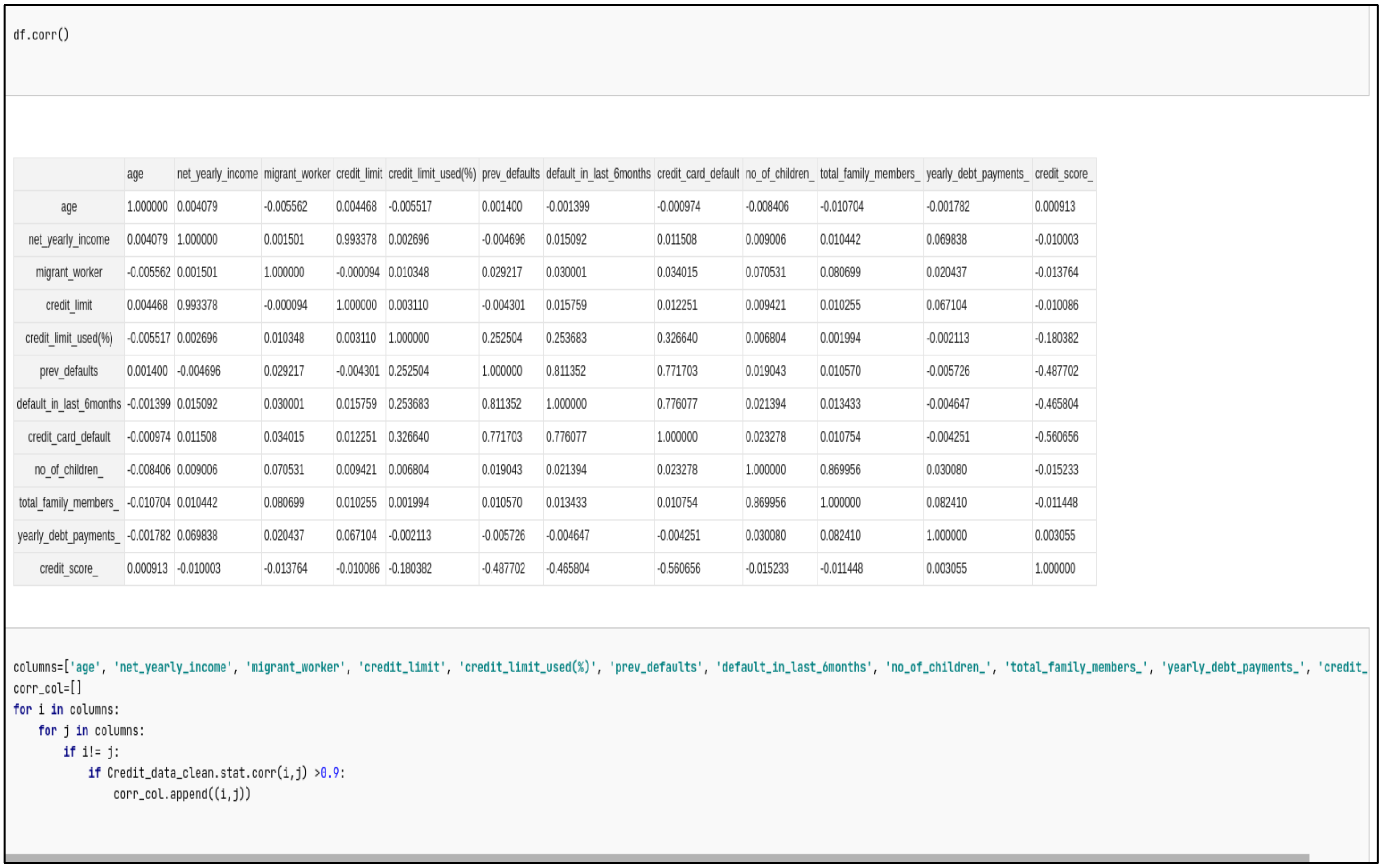
To replace the outliers, we have calculated the mean and the standard deviation of the respective columns and replaced all the values which exceed than the summation of mean and three times the standard deviation. Following is the code snippet.



```
from pyspark.sql.functions import mean as _mean, stddev as _stddev, col


columns=[ "net_yearly_income", "credit_limit", "credit_limit_used(%)", "yearly_debt_payments_"]

for c in columns:
    df_stats = Credit_data_imputed.select(
                _mean(col(c)).alias('mean'),
                _stddev(col(c)).alias('std')
            ).collect()

    mean = df_stats[0]['mean']
    std = df_stats[0]['std']

    Credit_data_clean = Credit_data_imputed.withColumn(c, (when(Credit_data_imputed[c]>(mean+(3*std)),mean).otherwise(Credit_data_imputed[c])))
```

Feature selection:

Feature Selection is the method of reducing the input variable to your model by using only relevant data and getting rid of noise in data. We have done the Feature Selection using Correlation. Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable.



```
df.corr()
```

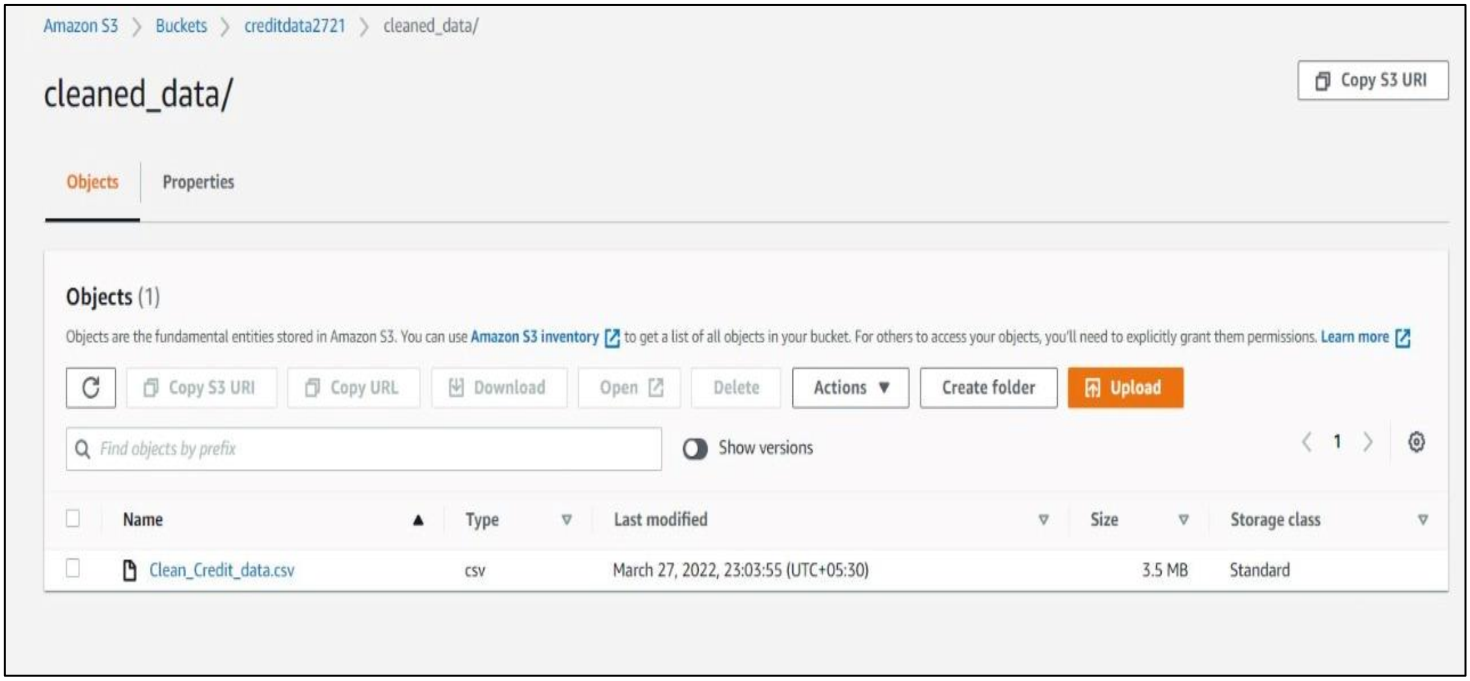
	age	net_yearly_income	migrant_worker	credit_limit	credit_limit_used(%)	prev_defaults	default_in_last_6months	credit_card_default	no_of_children_	total_family_members_	yearly_debt_payments_	credit_score_
age	1.000000	0.004079	-0.005562	0.004468	-0.005517	0.001400	-0.001399	-0.000974	-0.008406	-0.010704	-0.001782	0.000913
net_yearly_income	0.004079	1.000000	0.001501	0.993378	0.002696	-0.004696	0.015092	0.011508	0.009006	0.010442	0.069838	-0.010003
migrant_worker	-0.005562	0.001501	1.000000	-0.000094	0.010348	0.029217	0.030001	0.034015	0.070531	0.080699	0.020437	-0.013764
credit_limit	0.004468	0.993378	-0.000094	1.000000	0.003110	-0.004301	0.015759	0.012251	0.009421	0.010255	0.067104	-0.010086
credit_limit_used(%)	-0.005517	0.002696	0.010348	0.003110	1.000000	0.252504	0.253683	0.326640	0.006804	0.001994	-0.002113	-0.180382
prev_defaults	0.001400	-0.004696	0.029217	-0.004301	0.252504	1.000000	0.811352	0.771703	0.019043	0.010570	-0.005726	-0.487702
default_in_last_6months	-0.001399	0.015092	0.030001	0.015759	0.253683	0.811352	1.000000	0.776077	0.021394	0.013433	-0.004647	-0.465804
credit_card_default	-0.000974	0.011508	0.034015	0.012251	0.326640	0.771703	0.776077	1.000000	0.023278	0.010754	-0.004251	-0.560656
no_of_children_	-0.008406	0.009006	0.070531	0.009421	0.006804	0.019043	0.021394	0.023278	1.000000	0.869956	0.030080	-0.015233
total_family_members_	-0.010704	0.010442	0.080699	0.010255	0.001994	0.010570	0.013433	0.010754	0.869956	1.000000	0.082410	-0.011448
yearly_debt_payments_	-0.001782	0.069838	0.020437	0.067104	-0.002113	-0.005726	-0.004647	-0.004251	0.030080	0.082410	1.000000	0.003055
credit_score_	0.000913	-0.010003	-0.013764	-0.010086	-0.180382	-0.487702	-0.465804	-0.560656	-0.015233	-0.011448	0.003055	1.000000

```
columns=['age', 'net_yearly_income', 'migrant_worker', 'credit_limit', 'credit_limit_used(%)', 'prev_defaults', 'default_in_last_6months', 'no_of_children_', 'total_family_members_', 'yearly_debt_payments_', 'credit_
corr_col=[]
for i in columns:
    for j in columns:
        if i!= j:
            if Credit_data_clean.stat.corr(i,j) >0.9:
                corr_col.append((i,j))
```

7. Data Analysis:

A PowerBI Dashboard is a single page visualization with multiple charts and graphs to tell a story. A power BI dashboard enables users to analyze reports and view all important metrics at a glance. Using a Power BI dashboard, users can create visualizations from multiple datasets or multiple reports It can be embedded into applications to provide a unified user experience and can also be shared within organization. We have built a Dashboard from the Cleaned Data using Power BI.

We have loaded the data into Power BI by integrating it to Amazon S3 Bucket, where the Cleaned Data is a Saved. Below is the code snippet of it



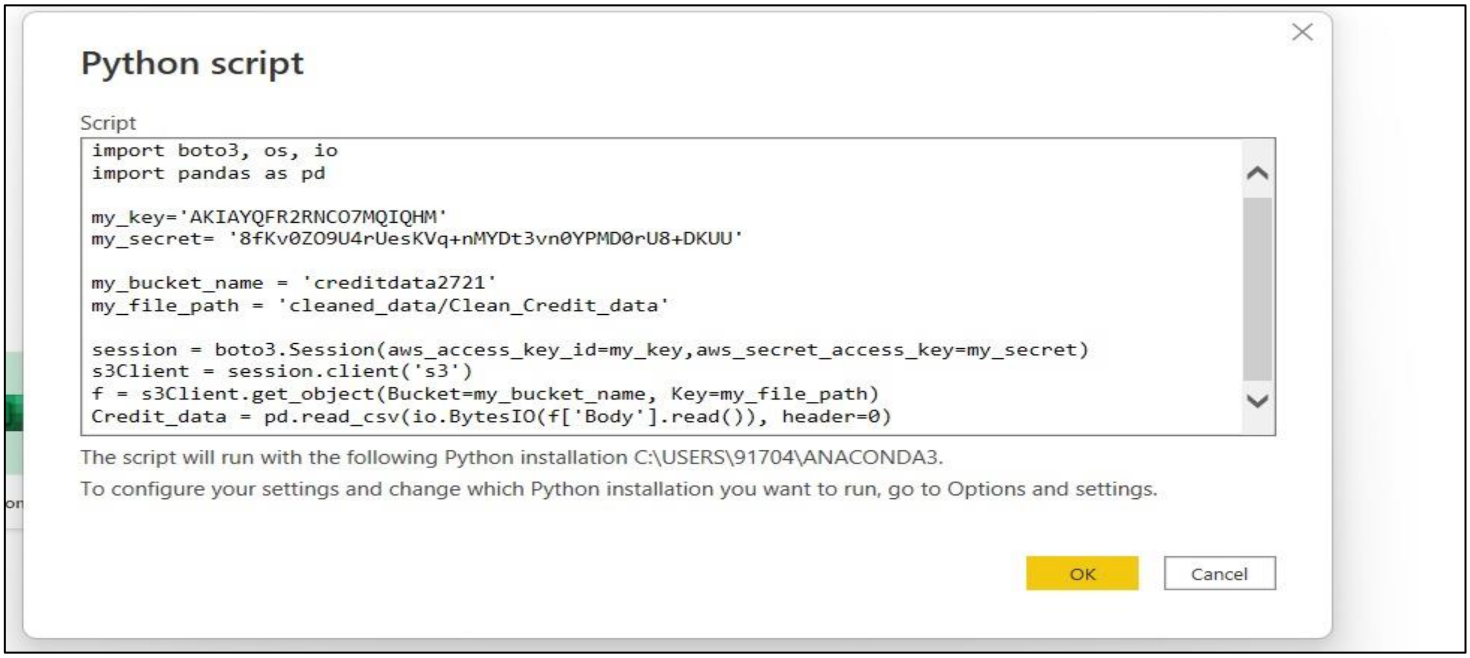
cleaned_data/

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name ▲	Type ▽	Last modified ▽	Size ▽	Storage class ▽
□	Clean_Credit_data.csv	csv	March 27, 2022, 23:03:55 (UTC+05:30)	3.5 MB	Standard

Below is the code snippet for integrating Power BI with AWS S3 bucket

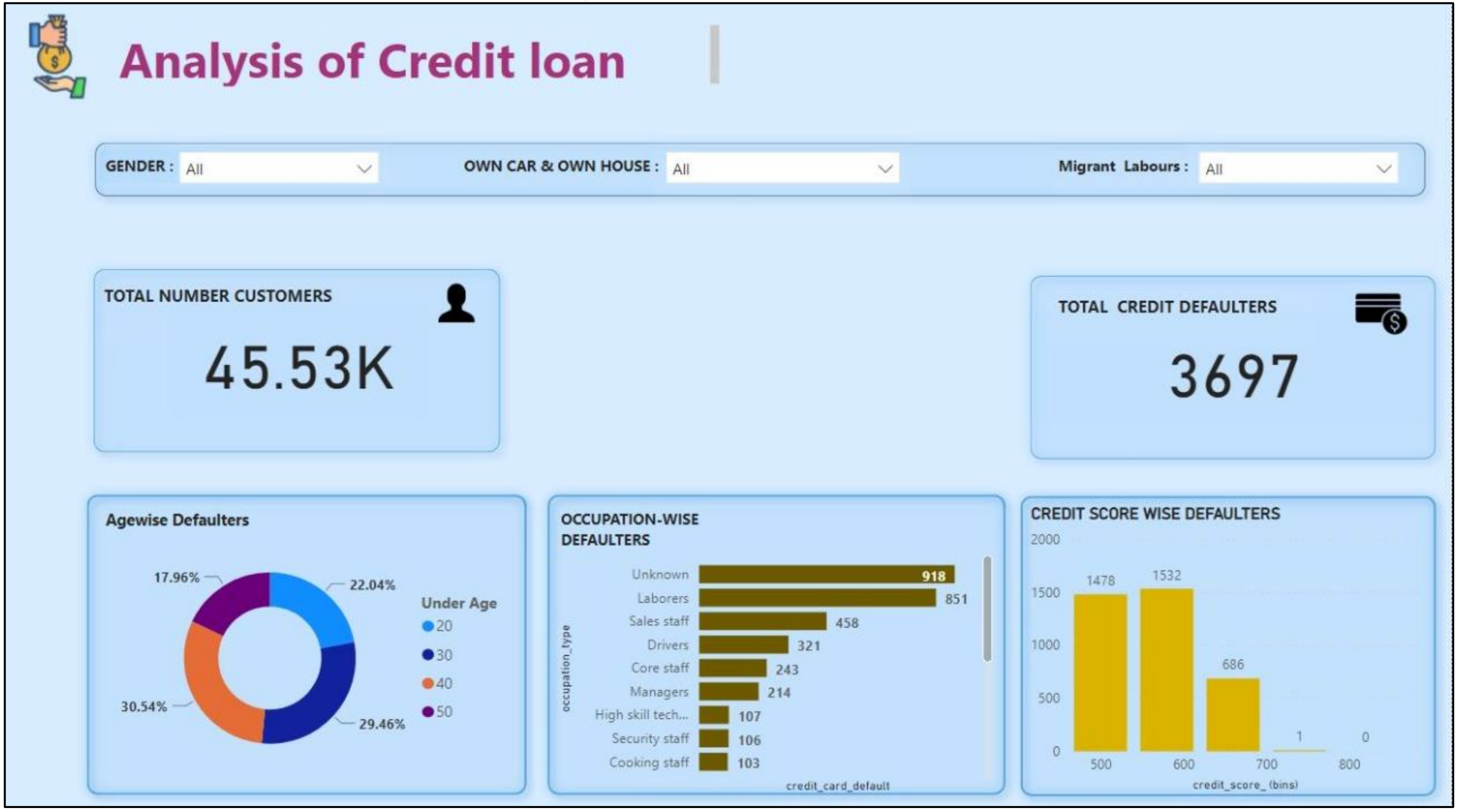


```
import boto3, os, io
import pandas as pd

my_key='AKIAYQFR2RNCO7MQIQHM'
my_secret= '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'

my_bucket_name = 'creditdata2721'
my_file_path = 'cleaned_data/Clean_Credit_data'

session = boto3.Session(aws_access_key_id=my_key,aws_secret_access_key=my_secret)
s3Client = session.client('s3')
f = s3Client.get_object(Bucket=my_bucket_name, Key=my_file_path)
Credit_data = pd.read_csv(io.BytesIO(f['Body'].read()), header=0)
```

The script will run with the following Python installation C:\USERS\91704\ANACONDA3.

To configure your settings and change which Python installation you want to run, go to Options and settings.

Below is Dashboard for ‘Analysis of Credit loan’ defaulters. It is categorized on the gender, whether a person owns a house & car and if a person is Migrant Labour.



Analysis of Credit loan

GENDER : All

OWN CAR & OWN HOUSE : All

Migrant Labours : All

TOTAL NUMBER CUSTOMERS 45.53K

TOTAL CREDIT DEFAULTERS 3697

8. ML Modelling

These are the following models which we have used in our project

Stacking

Stacking is a way to ensemble multiple classifications or regression model. There are many ways to ensemble models, the widely known models are *Bagging* or *Boosting*. Bagging allows multiple similar models with high variance are averaged to decrease variance. Boosting builds multiple incremental models to decrease the bias, while keeping variance small.

Following models are used for stacking classifier:
Logistic Regression
K-Nearest Neighbor
Random Forest
Decision Tree Classifier

Voting

A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output. It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting.

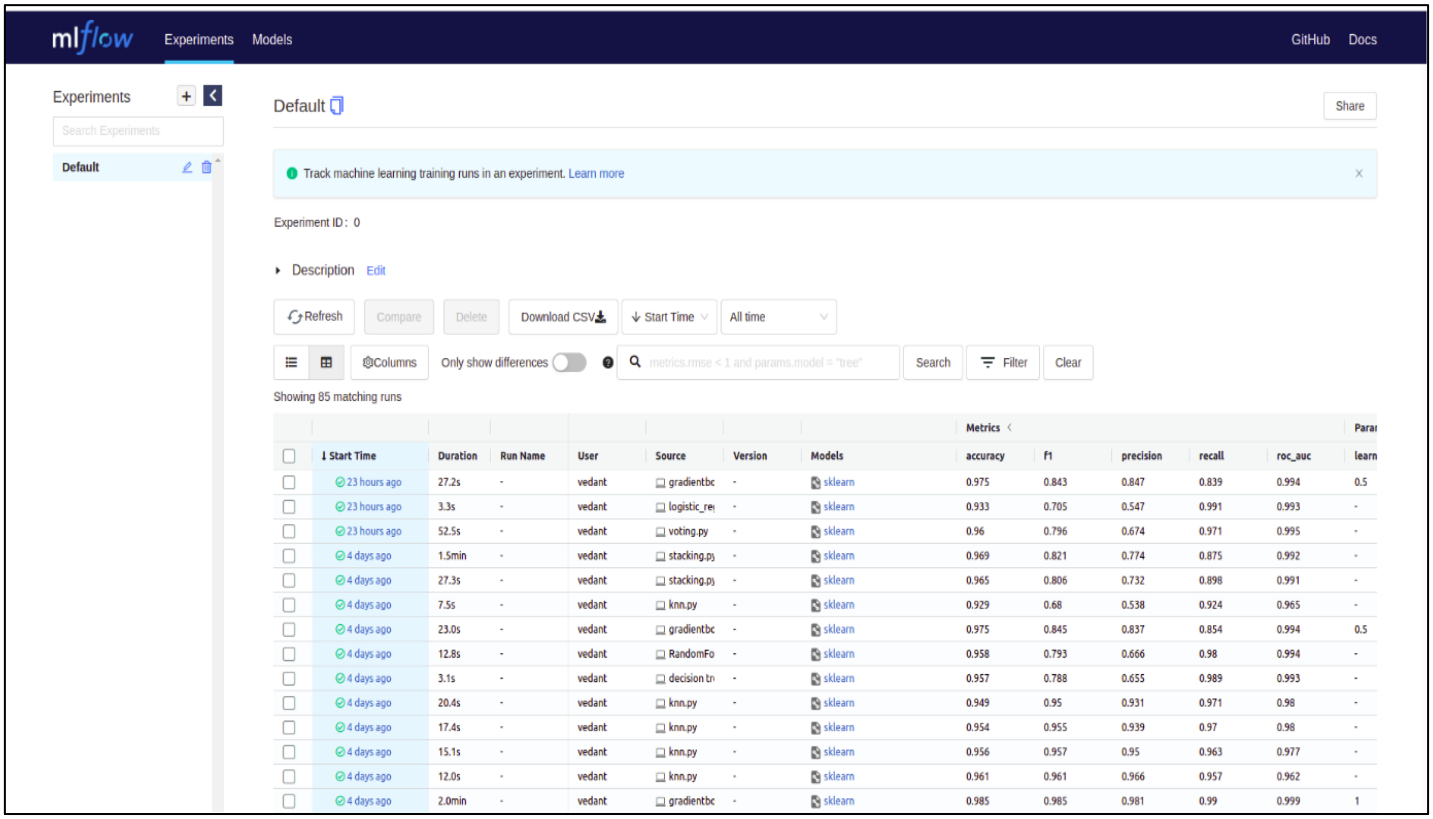
Following models are used for Voting classifier:
Logistic Regression
K-Nearest Neighbor
Random Forest
Decision Tree Classifier

Random Forest

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn't depend on one decision tree but multiple decision trees.

Gradient boosting

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.



Default

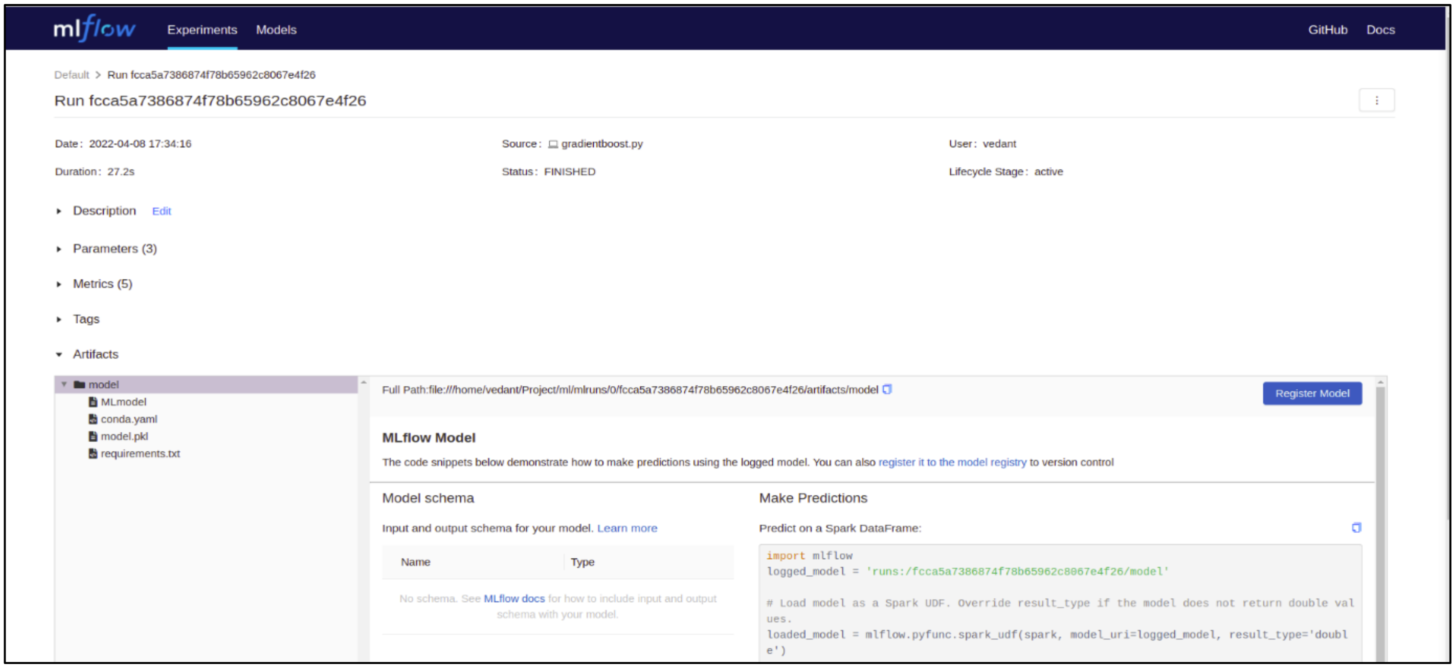
Track machine learning training runs in an experiment. Learn more

Experiment ID: 0

Description Edit

Showing 85 matching runs

□								Metrics					Parar
□	Start Time	Duration	Run Name	User	Source	Version	Models	accuracy	f1	precision	recall	roc_auc	learn
□	23 hours ago	27.2s	-	vedant	gradientbc	-	sklearn	0.975	0.843	0.847	0.839	0.994	0.5
□	23 hours ago	3.3s	-	vedant	logistic_re	-	sklearn	0.933	0.705	0.547	0.991	0.993	-
□	23 hours ago	52.5s	-	vedant	voting.py	-	sklearn	0.96	0.796	0.674	0.971	0.995	-
□	4 days ago	1.5min	-	vedant	stacking.py	-	sklearn	0.969	0.821	0.774	0.875	0.992	-
□	4 days ago	27.3s	-	vedant	stacking.py	-	sklearn	0.965	0.806	0.732	0.898	0.991	-
□	4 days ago	7.5s	-	vedant	knn.py	-	sklearn	0.929	0.68	0.538	0.924	0.965	-
□	4 days ago	23.0s	-	vedant	gradientbc	-	sklearn	0.975	0.845	0.837	0.854	0.994	0.5
□	4 days ago	12.8s	-	vedant	RandomFo	-	sklearn	0.958	0.793	0.666	0.98	0.994	-
□	4 days ago	3.1s	-	vedant	decision tr	-	sklearn	0.957	0.788	0.655	0.989	0.993	-
□	4 days ago	20.4s	-	vedant	knn.py	-	sklearn	0.949	0.95	0.931	0.971	0.98	-
□	4 days ago	17.4s	-	vedant	knn.py	-	sklearn	0.954	0.955	0.939	0.97	0.98	-
□	4 days ago	15.1s	-	vedant	knn.py	-	sklearn	0.956	0.957	0.95	0.963	0.977	-
□	4 days ago	12.0s	-	vedant	knn.py	-	sklearn	0.961	0.961	0.966	0.957	0.962	-
□	4 days ago	2.0min	-	vedant	gradientbc	-	sklearn	0.985	0.985	0.981	0.99	0.999	1



mlflow Experiments Models GitHub Docs

Default > Run fcca5a7386874f78b65962c8067e4f26

Run fcca5a7386874f78b65962c8067e4f26

Date: 2022-04-08 17:34:16 Source: gradientboost.py User: vedant

Duration: 27.2s Status: FINISHED Lifecycle Stage: active

- Description Edit
- Parameters (3)
- Metrics (5)
- Tags
- Artifacts

model
- MLmodel
- conda.yaml
- model.pkl
- requirements.txt

Full Path:file:///home/vedant/Project/ml/mlruns/0/fcca5a7386874f78b65962c8067e4f26/artifacts/model

Register Model

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. You can also register it to the model registry to version control

Model schema

Input and output schema for your model. Learn more

Name	Type

No schema. See MLflow docs for how to include input and output schema with your model.

Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
logged_model = 'runs:/fcca5a7386874f78b65962c8067e4f26/model'

# Load model as a Spark UDF. Override result_type if the model does not return double val
ues.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model, result_type='doubl
e')
```

Evaluation of the Model

Following are the metrics used for evaluating above models

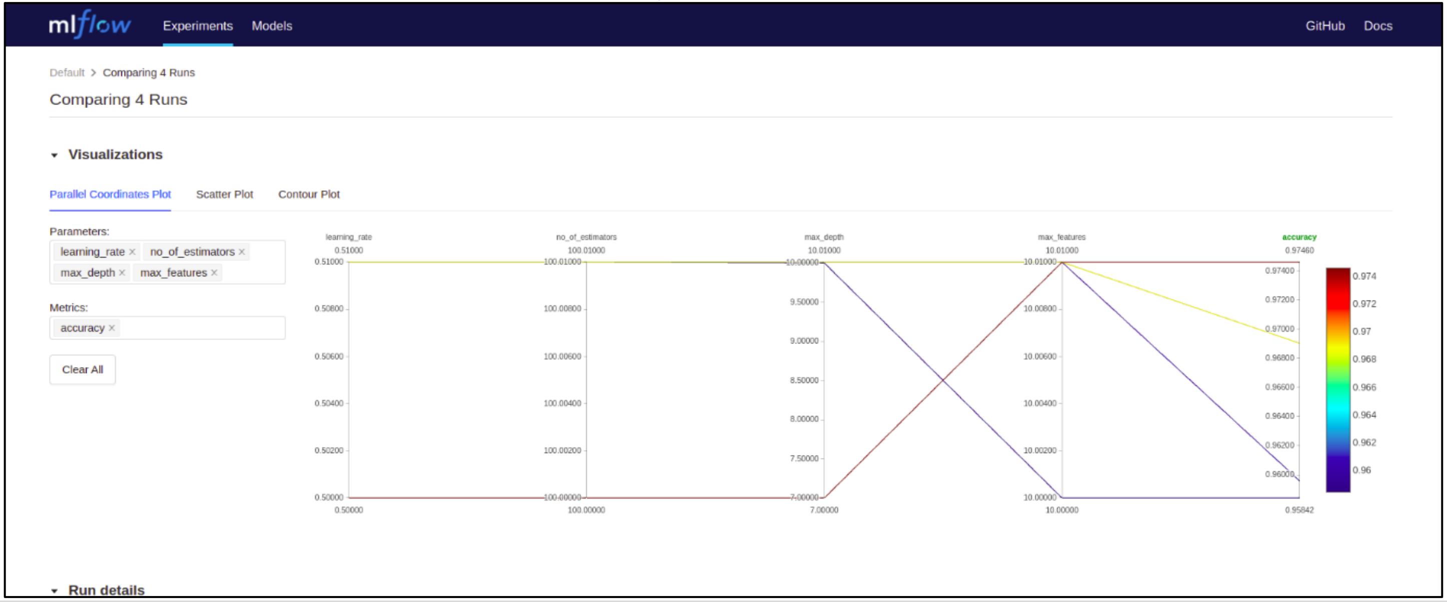
Accuracy Score

ROC-AUC Score

Precision Score

Recall Score

Below is the plot of Accuracy score for all of the above models



mlflow Experiments Models GitHub Docs

Default > Comparing 4 Runs

Comparing 4 Runs

- Visualizations

Parallel Coordinates Plot Scatter Plot Contour Plot

Parameters: learning_rate × no_of_estimators × max_depth × max_features ×

Metrics: accuracy ×

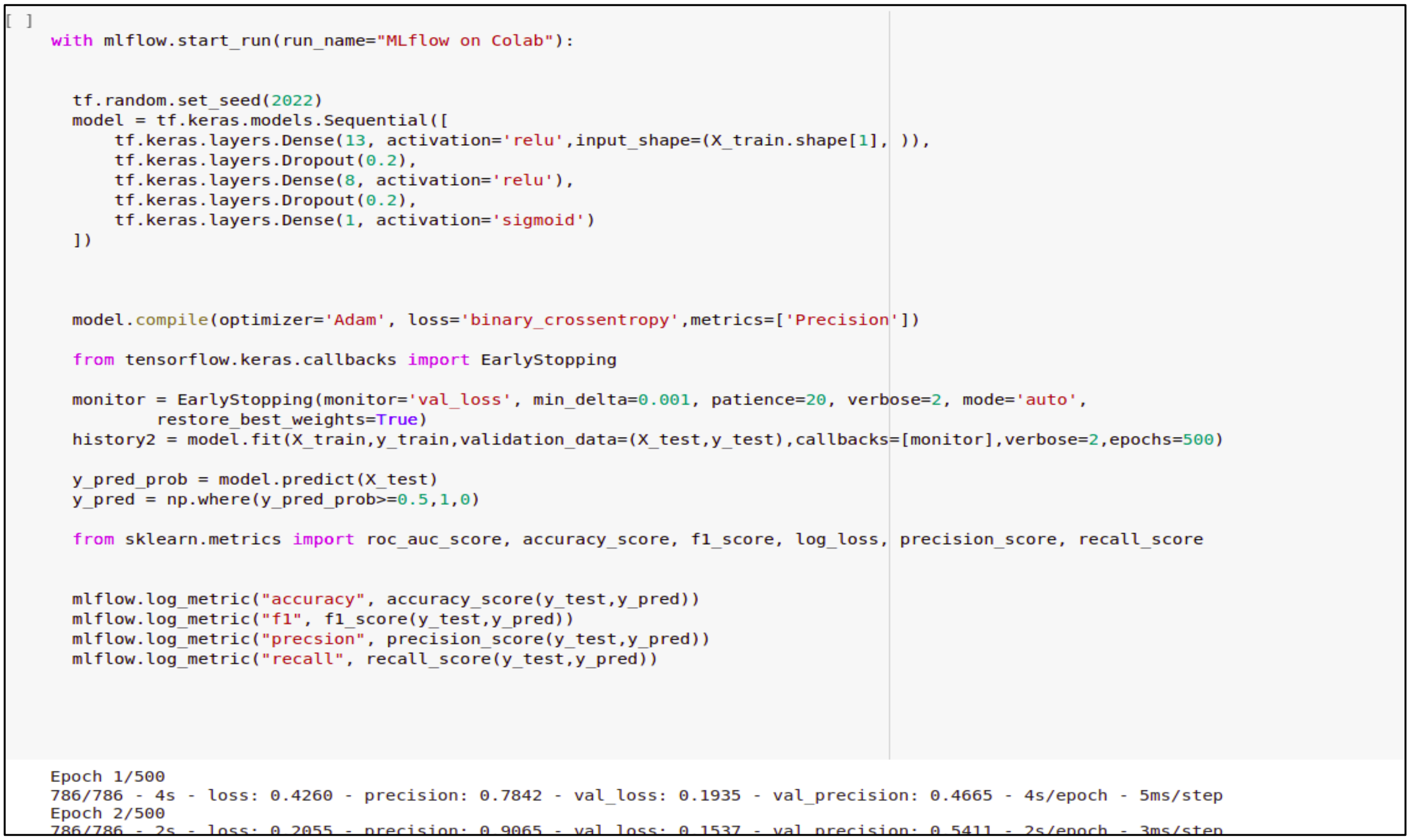
Clear All

- Run details

ANN

The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modeled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes. The term "Artificial Neural Network" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.

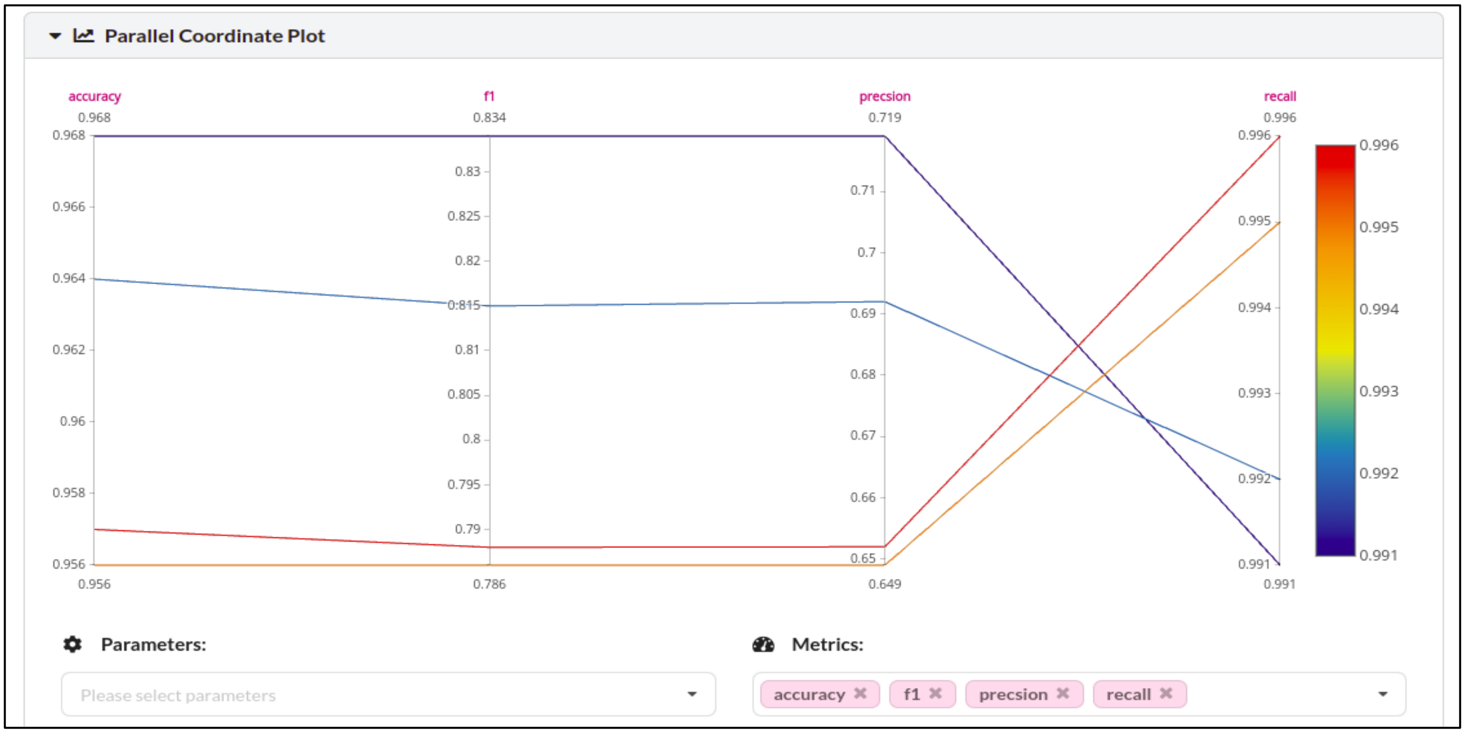
Below is the code snippet of ANN model



```
[ ]
    with mlflow.start_run(run_name="MLflow on Colab"):


      tf.random.set_seed(2022)
      model = tf.keras.models.Sequential([
          tf.keras.layers.Dense(13, activation='relu',input_shape=(X_train.shape[1], )),
          tf.keras.layers.Dropout(0.2),
          tf.keras.layers.Dense(8, activation='relu'),
          tf.keras.layers.Dropout(0.2),
          tf.keras.layers.Dense(1, activation='sigmoid')
      ])



      model.compile(optimizer='Adam', loss='binary_crossentropy',metrics=['Precision'])

      from tensorflow.keras.callbacks import EarlyStopping

      monitor = EarlyStopping(monitor='val_loss', min_delta=0.001, patience=20, verbose=2, mode='auto',
              restore_best_weights=True)
      history2 = model.fit(X_train,y_train,validation_data=(X_test,y_test),callbacks=[monitor],verbose=2,epochs=500)

      y_pred_prob = model.predict(X_test)
      y_pred = np.where(y_pred_prob>=0.5,1,0)

      from sklearn.metrics import roc_auc_score, accuracy_score, f1_score, log_loss, precision_score, recall_score


      mlflow.log_metric("accuracy", accuracy_score(y_test,y_pred))
      mlflow.log_metric("f1", f1_score(y_test,y_pred))
      mlflow.log_metric("precsion", precision_score(y_test,y_pred))
      mlflow.log_metric("recall", recall_score(y_test,y_pred))
```

```
Epoch 1/500
786/786 - 4s - loss: 0.4260 - precision: 0.7842 - val_loss: 0.1935 - val_precision: 0.4665 - 4s/epoch - 5ms/step
Epoch 2/500
786/786 - 2s - loss: 0.2055 - precision: 0.9065 - val_loss: 0.1537 - val_precision: 0.5411 - 2s/epoch - 3ms/step
```

Below is the evaluation metrics (Accuracy Score) of ANN



Parameters:

Please select parameters

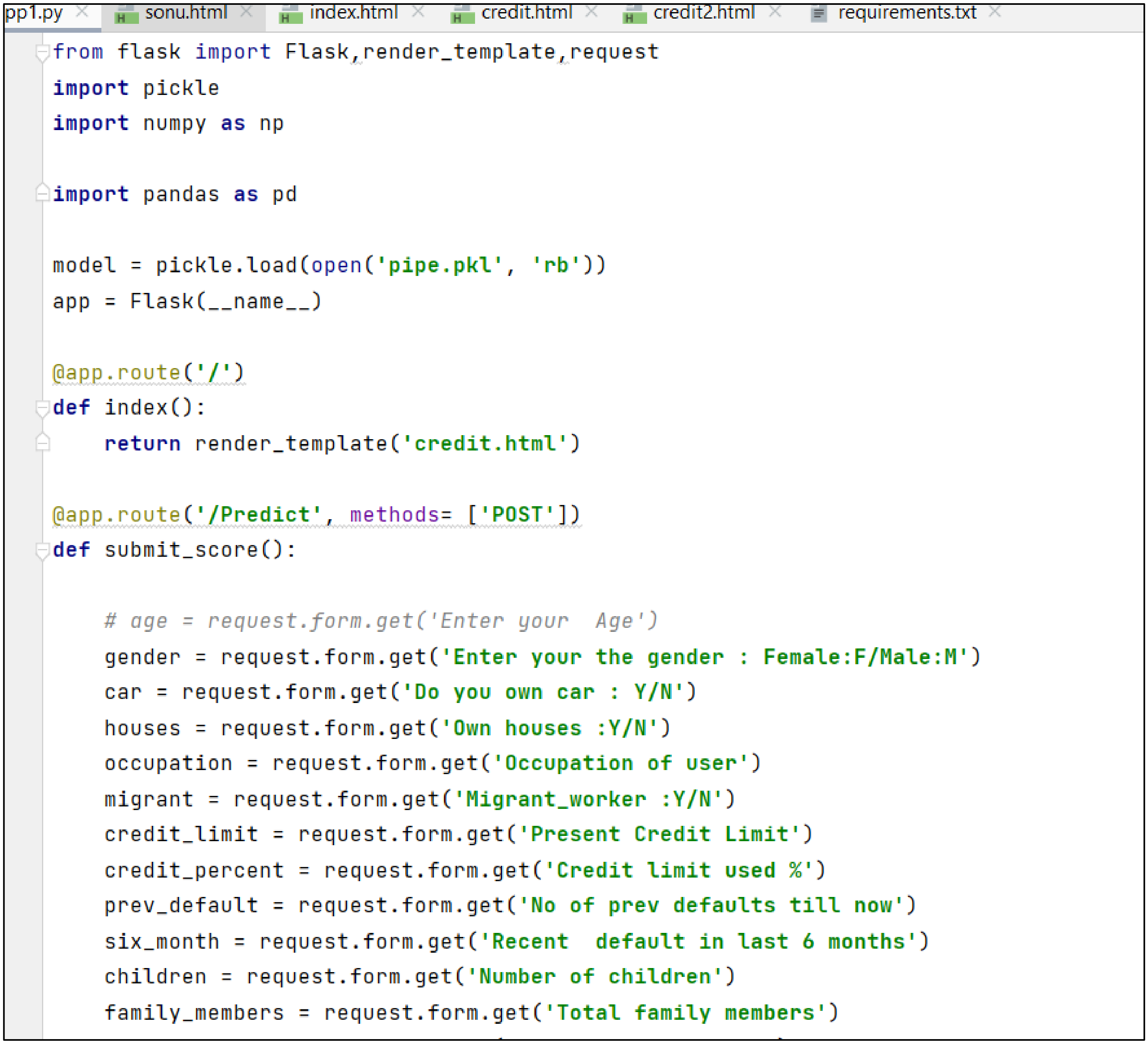
Metrics:

accuracy ✖ f1 ✖ precsion ✖ recall ✖

9. Deploying the Model on Webserver:

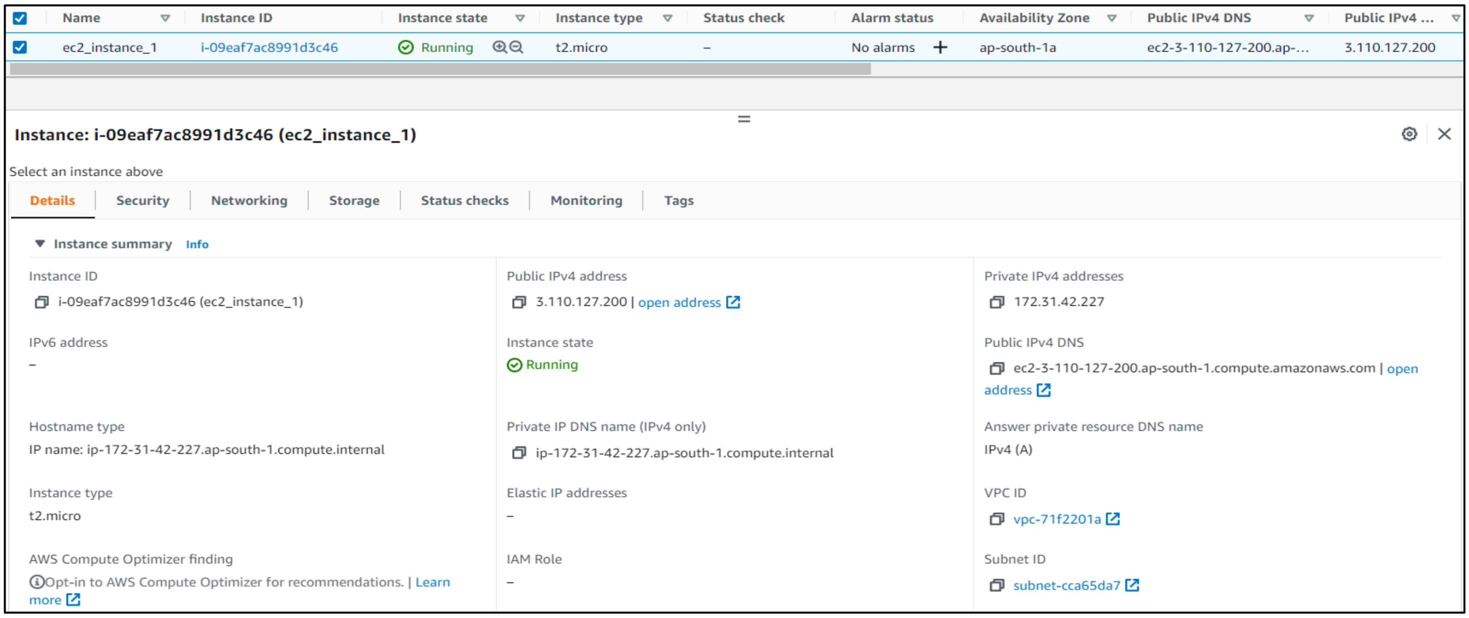
For deploying the model on server, we have used flask to create a Web Application and then deploy it on AWS EC2 Instance Webserver.

Below is the code snippet for building web application using Flask



```python
from flask import Flask,render_template,request
import pickle
import numpy as np

import pandas as pd

model = pickle.load(open('pipe.pkl', 'rb'))
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('credit.html')

@app.route('/Predict', methods= ['POST'])
def submit_score():

    # age = request.form.get('Enter your  Age')
    gender = request.form.get('Enter your the gender : Female:F/Male:M')
    car = request.form.get('Do you own car : Y/N')
    houses = request.form.get('Own houses :Y/N')
    occupation = request.form.get('Occupation of user')
    migrant = request.form.get('Migrant_worker :Y/N')
    credit_limit = request.form.get('Present Credit Limit')
    credit_percent = request.form.get('Credit limit used %')
    prev_default = request.form.get('No of prev defaults till now')
    six_month = request.form.get('Recent  default in last 6 months')
    children = request.form.get('Number of children')
    family_members = request.form.get('Total family members')
```

Following is the Amazon EC2 webserver instance used for deploying the application



☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
☑	ec2_instance_1	i-09eaf7ac8991d3c46	✓ Running	t2.micro	–	No alarms +	ap-south-1a	ec2-3-110-127-200.ap-...	3.110.127.200

Instance: i-09eaf7ac8991d3c46 (ec2_instance_1)

Select an instance above

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

▼ Instance summary Info

Instance ID
i-09eaf7ac8991d3c46 (ec2_instance_1)

Public IPv4 address
3.110.127.200 | open address

Private IPv4 addresses
172.31.42.227

IPv6 address
–

Instance state
✓ Running

Public IPv4 DNS
ec2-3-110-127-200.ap-south-1.compute.amazonaws.com | open address

Hostname type
IP name: ip-172-31-42-227.ap-south-1.compute.internal

Private IP DNS name (IPv4 only)
ip-172-31-42-227.ap-south-1.compute.internal

Answer private resource DNS name
IPv4 (A)

Instance type
t2.micro

Elastic IP addresses
–

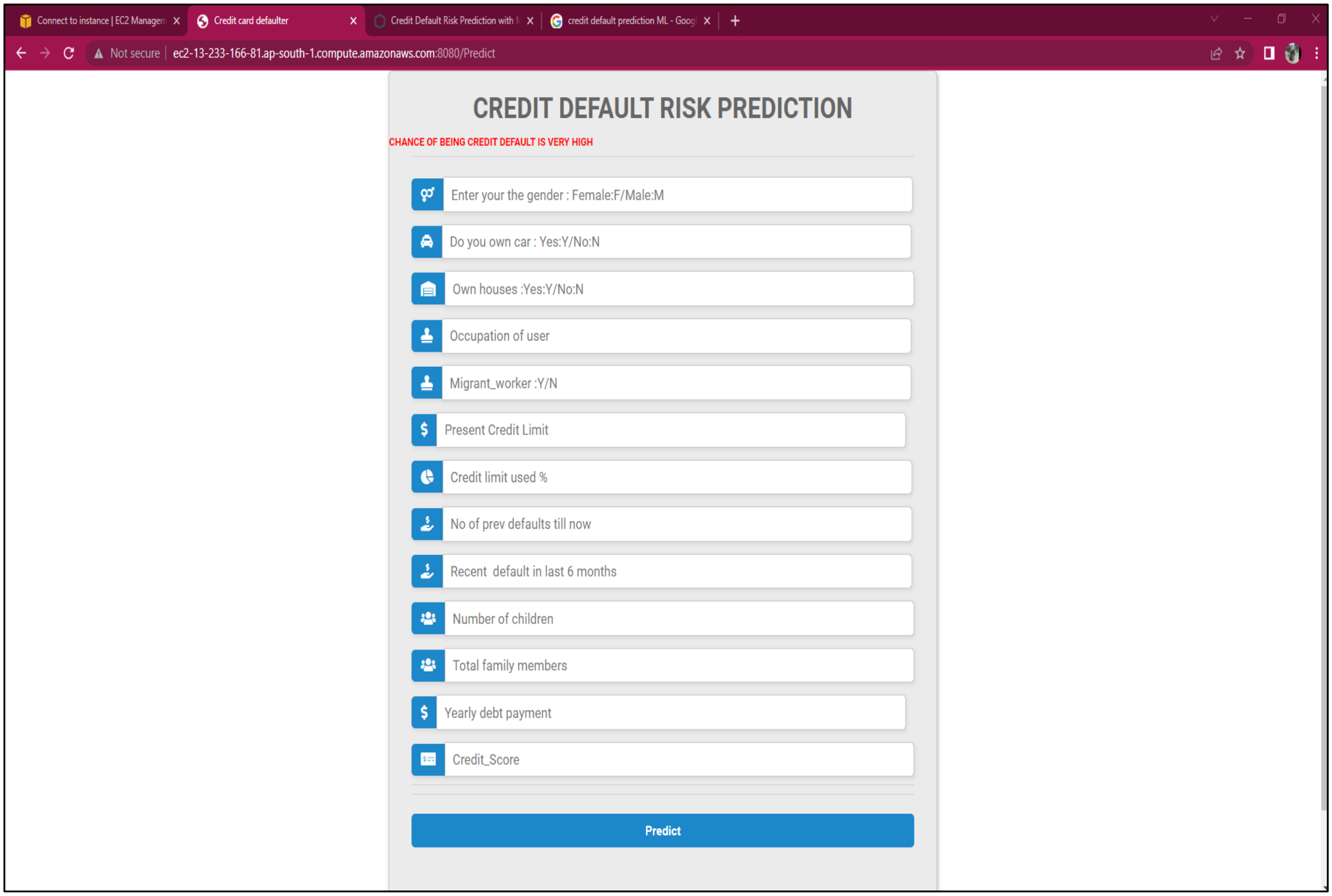
VPC ID
vpc-71f2201a

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more

IAM Role
–

Subnet ID
subnet-cca65da7

Following is the webpage where we have to enter the details like gender, the Credit limit used, Present Credit limit, number of previous defaults etc. The model collects the given input and predicts if the user will be a defaulter or not.


Connect to instance | EC2 Managem
Credit card defaulter
Credit Default Risk Prediction with M
credit default prediction ML - Goog
Not secure
ec2-13-233-166-81.ap-south-1.compute.amazonaws.com:8080/Predict

CREDIT DEFAULT RISK PREDICTION

CHANCE OF BEING CREDIT DEFAULT IS VERY HIGH

Enter your the gender : Female:F/Male:M

Do you own car : Yes:Y/No:N

Own houses :Yes:Y/No:N

Occupation of user

Migrant_worker :Y/N

Present Credit Limit

Credit limit used %

No of prev defaults till now

Recent default in last 6 months

Number of children

Total family members

Yearly debt payment

Credit_Score

10. Conclusion

In conclusion, we provide an automated model which predicts a potential credit loan defaulter which we are generating by fetching data of bank customers and performing classification machine learning algorithms on it.

11.References

- https://www.kaggle.com/
- https://www.analyticsvidhya.com/
- https://www.mlflow.org/docs/latest/index.html
- https://spark.apache.org/docs/latest/
- https://docs.microsoft.com/en-us/power-bi/
- https://scikit-learn.org/stable/
- https://flask.palletsprojects.com/en/2.1.x/
- https://towardsdatascience.com/
- https://medium.com/