Final Project

Data Science Course

presented by **Power Ranger Team**

Our Team

Fajar

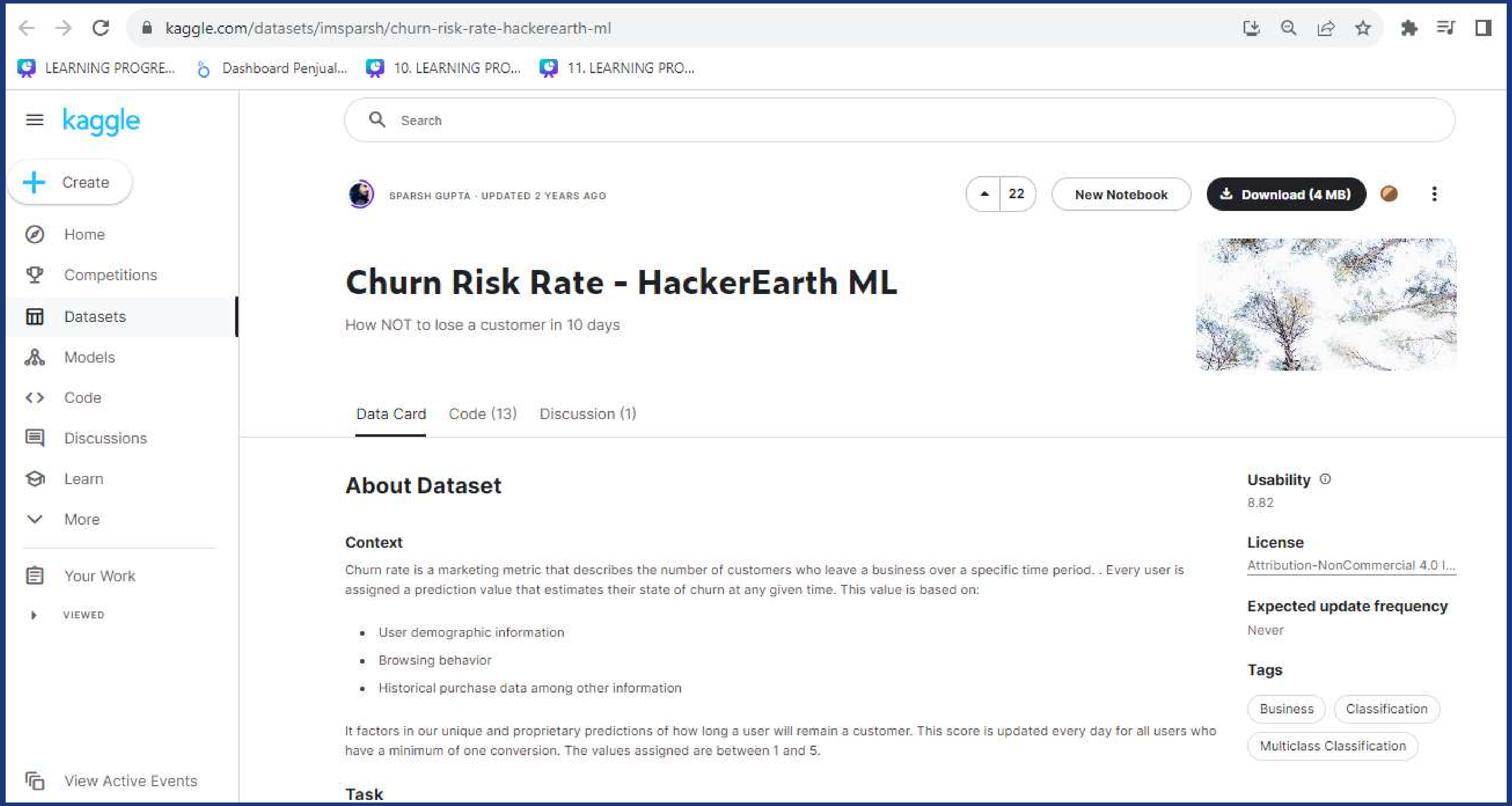
Melani

Power Ranger

Contents

Data Understanding

Dataset



Source (kaggle): https://www.kaggle.com/datasets/imsparsh/churn-risk-rate-hackerearth-ml

Dataset Churn Risk Rate ini kami dapatkan dari website kaggle, merupakan dataset yang di unggah oleh akun Sparsh Gupta. Dataset Churn Risk Rate ini merupakan dataset milik perusahaan HackerEarth yang bergerak dalam bidang tech dari India yang berkantor di Amerika, perusahaan tersebut menyediakan jasa untuk developing proyek perangkat lunak atau solusi teknologi menggunakan machine learning.

Dataset

Latar Belakang:
Dataset Churn Risk ini adalah data behaviour dari masing-masing customer yang churn/tidak churn dalam penggunaan akses pada sebuah website milik hackerearth, yang direpresentasikan dalam bentuk rate (-1 s/d 5). Semakin besar nilai rate churn maka semakinn besar kemungkinan customer tersebut akan meninggalkan menggunakan layanan akses website tersebut.

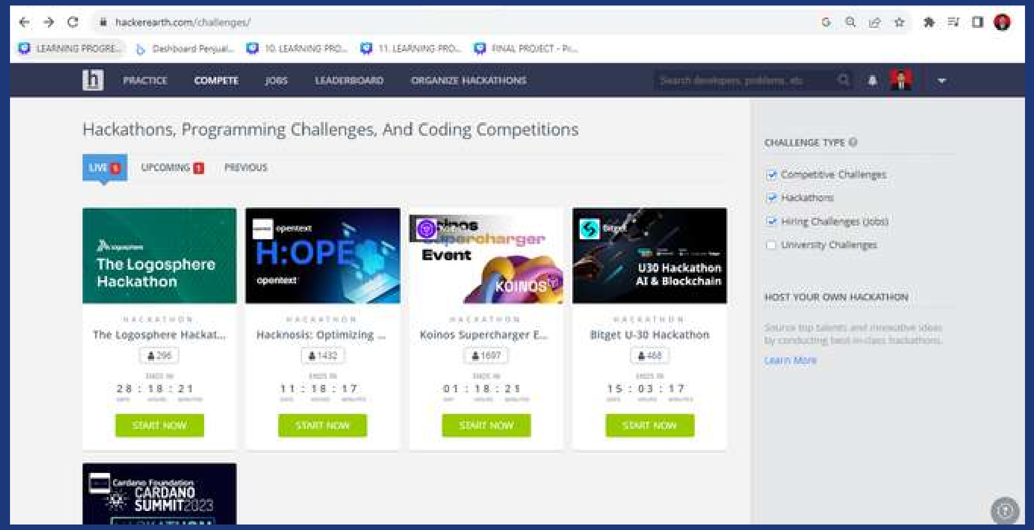
Dari dataset tersebut dapat dibuat sebuah model prediksi menggunakan machine learning untuk mengetahui customer mana yang akan tetap setia/ meninggalkan penggunaan layanan website tersebut

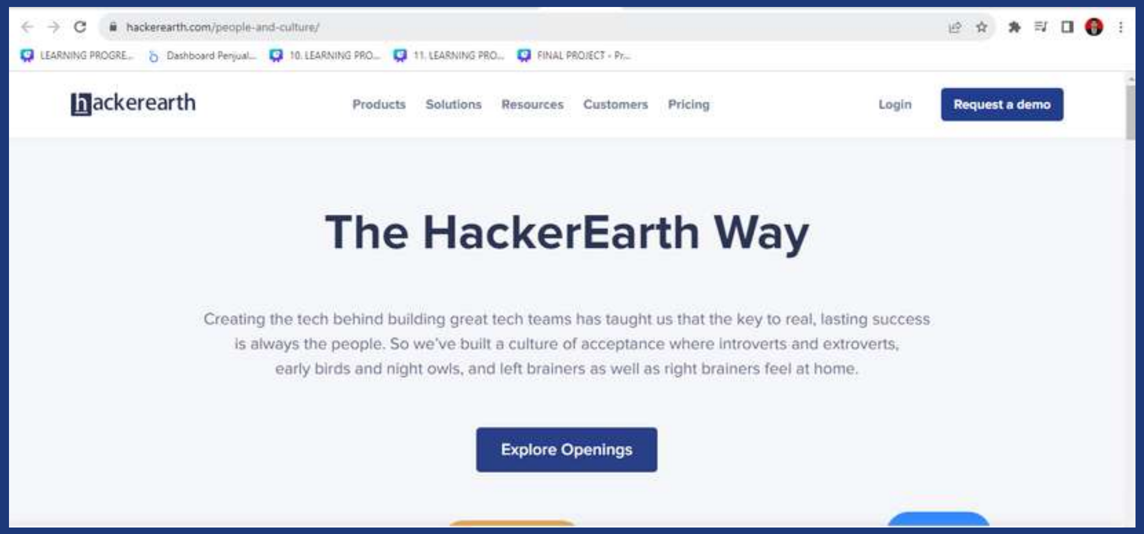
Tujuan Project:
Membuat model prediksi machine learning yang paling efektif untuk mengetahui apakah customer akan churn/no churn berdasarkan data behaviour customer dalam melakukan akses website milik hackerearth.

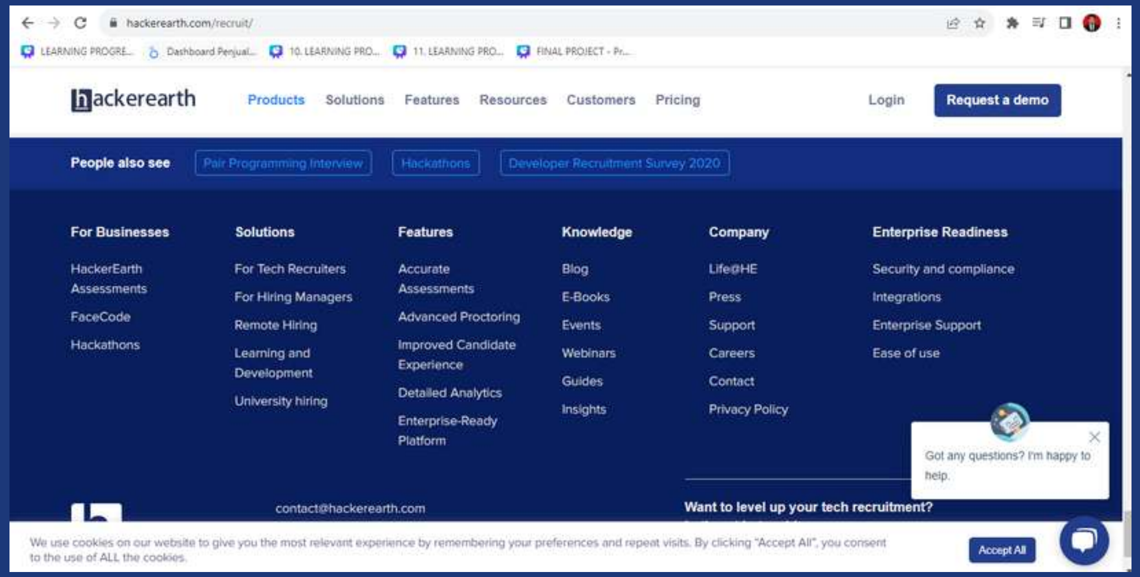
Business Understand

Apa itu web HackerEarth?

HackerEarth adalah perusahaan perangkat lunak India yang berkantor pusat di San Francisco, AS, yang menyediakan perangkat lunak perusahaan yang membantu organisasi dalam *technical hiring*. HackerEarth digunakan oleh organisasi untuk penilaian keterampilan teknis dan wawancara video jarak jauh. Selain itu HackerEarth juga menyediakan *assesment* bagi para *developers* untuk melatih skill dalam bidang pemrograman, karena perusahaan client mempekerjakan *developers* lebih cepat dengan HackerEarth *Assessments*.







Dataset Overview

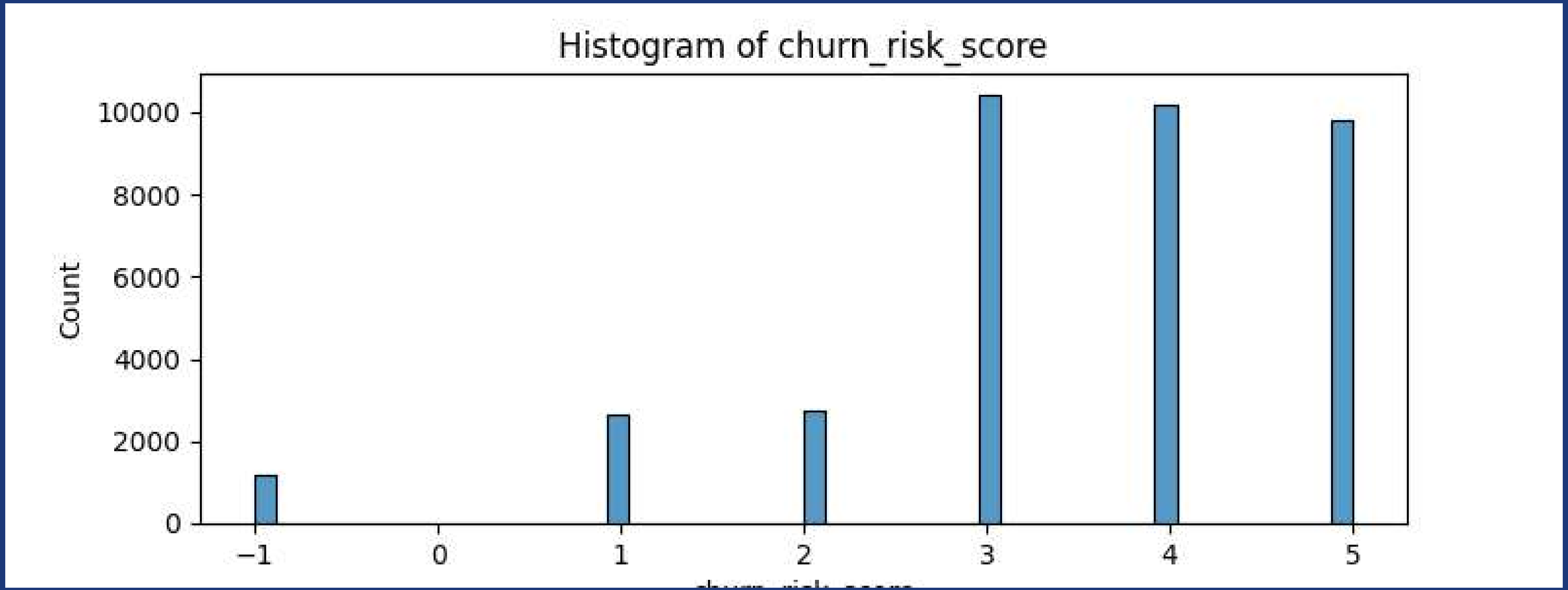
Nama kolom	Deskripsi	Tipe Data
customer_id	nomer identitas customer	Kategorikal
age	umur	Numerik
gender	jenis kelamin	Kategorikal
region_category	Mewakili wilayah tempat seorang pelanggan berasal.	Kategorikal
membership_category	kategori membership yang customer gunakan	Kategorikal
joined_through_referral	Mewakili apakah seorang pelanggan bergabung menggunakan kode atau ID referral apa pun.	Kategorikal
avg_time_spent	rata-rata waktu yang dihabiskan customer ketika mengakses website	Numerikal
avg_transaction_value	rata-rata nilai transaksi yang dilakukan oleh customer	Numerikal
avg_frequency_login_days	Mewakili jumlah kali seorang pelanggan telah masuk ke situs web.	Numerikal
points_in_wallet	Mewakili jumlah poin yang diberikan kepada seorang pelanggan setiap kali transaksi dilakukan.	Numerikal
used_special_discount	Mewakili apakah seorang pelanggan menggunakan diskon khusus yang ditawarkan.	Numerikal
offer_application_preference	Mewakili apakah seorang pelanggan lebih memilih tawaran	Boolean
past_complaint	Mewakili apakah seorang pelanggan telah mengajukan keluhan di masa lalu.	Boolean
complaint_status	Mewakili apakah keluhan yang diajukan oleh seorang pelanggan telah diselesaikan.	Kategorikal
feedback	Mewakili feedback yang diberikan oleh seorang pelanggan.	Kategorikal
churn_risk_score	Mewakili skor risiko pergantian pelanggan yang berkisar dari 1 hingga 5.	Numerikal

Sebelum memulai sebuah project kami mencoba untuk memahami fitur apa saja yang kami miliki, sehingga kami bisa mengambil wawasan dalam dataset churn risk score ini ada 3 jenis tipe data yaitu kategorikal, numerik, serta boolean.

selain itu kami juga memahami deskripsi value yang ada pada setiap fitur sehingga bisa lebih memahami dataset tersebut ketika akan melakukan visualisasi ataupun tahap pra-pemrosesan.

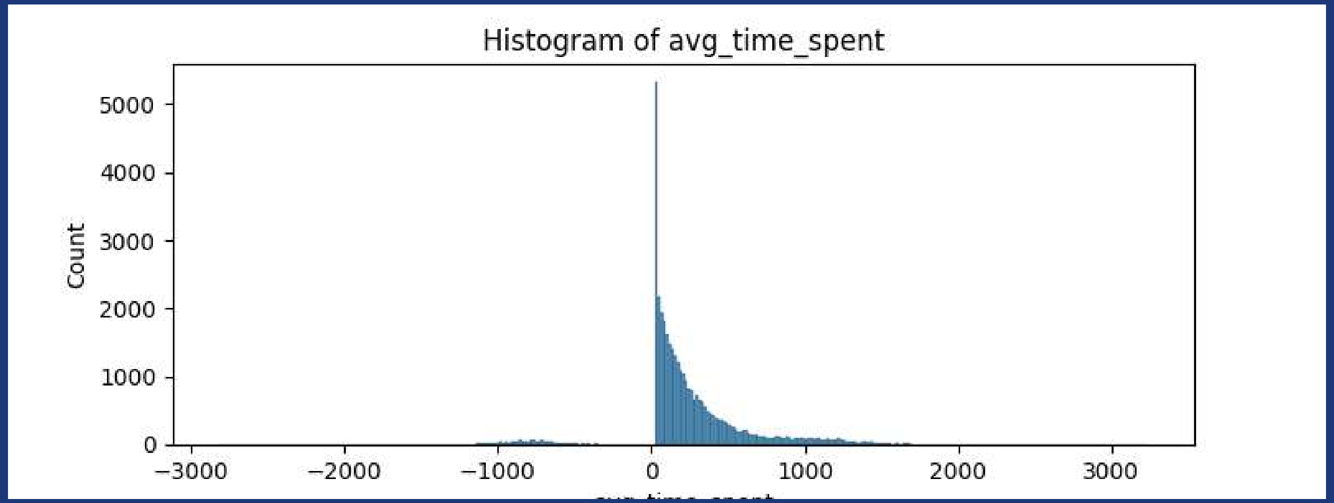
Exploration Data Analysis

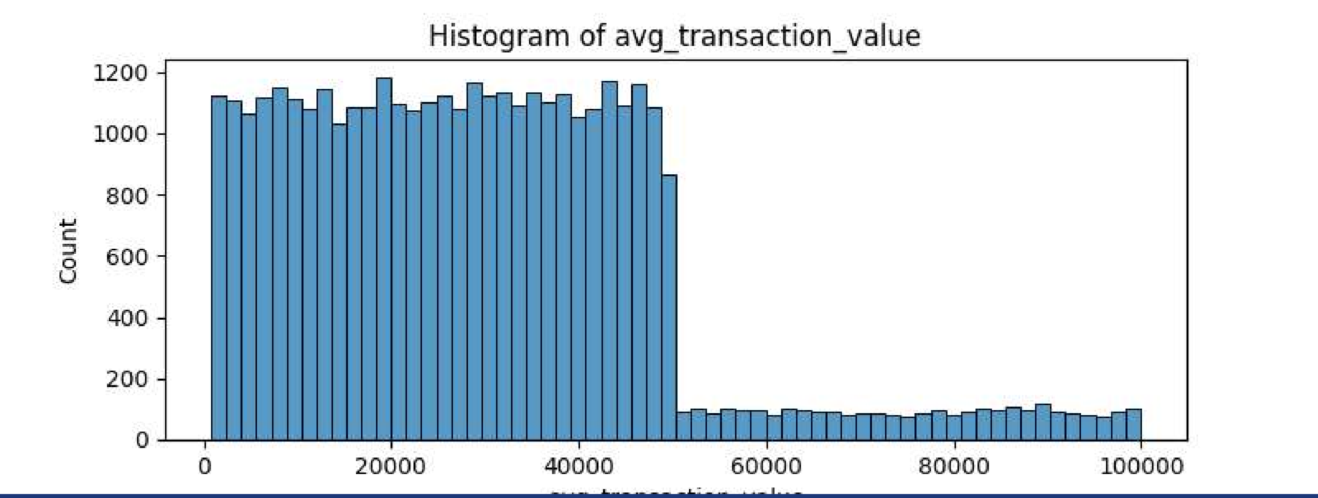
Distribusi variabel target

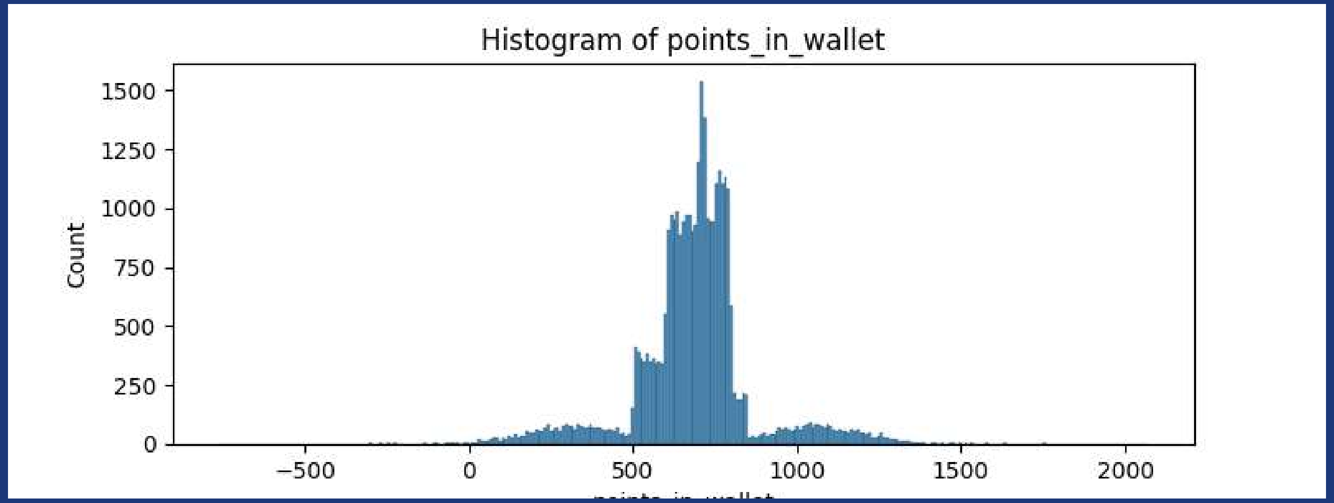


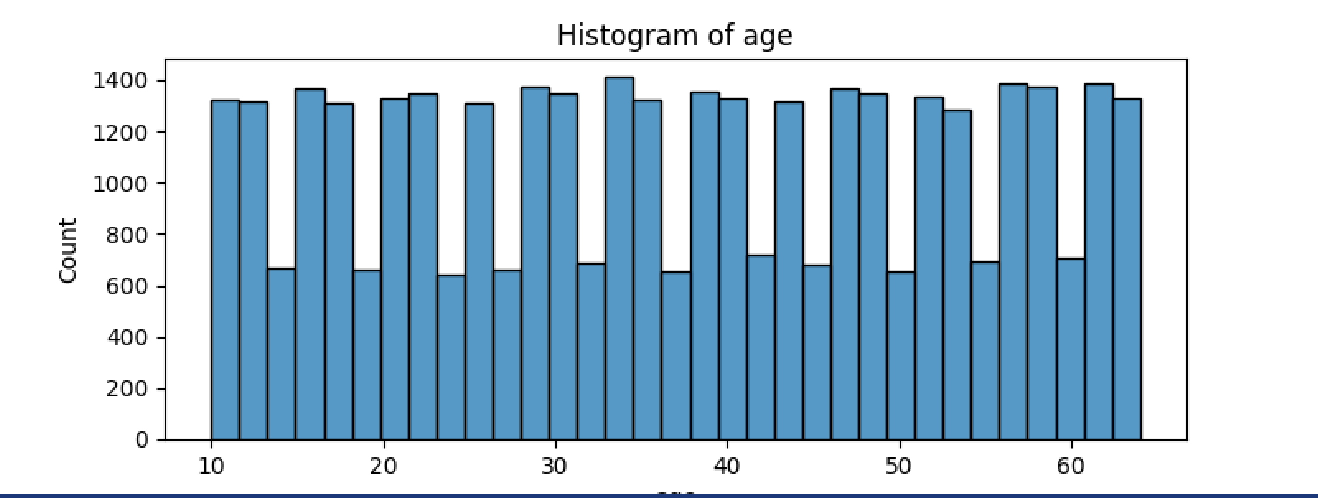
Plot diatas merupakan countplot dari variabel churn-rick_score, dari visusalisasi diatas terlihat bahawa churn_risk score memiliki rate dengan rentang -1 s/d 5.Pada kolom target terdapat data -1 yang akan diubah menjadi 1 sehingga tidak terdapat nilai "-" (minus), karena churn risk score -1 memiliki arti risiko untuk berhenti berlangganan (churn) rendah sehingga disatukan/digolongkan bersama nilai 1 saat melakukan pra pemrosesan nantinya.

Distribusi variabel numerik





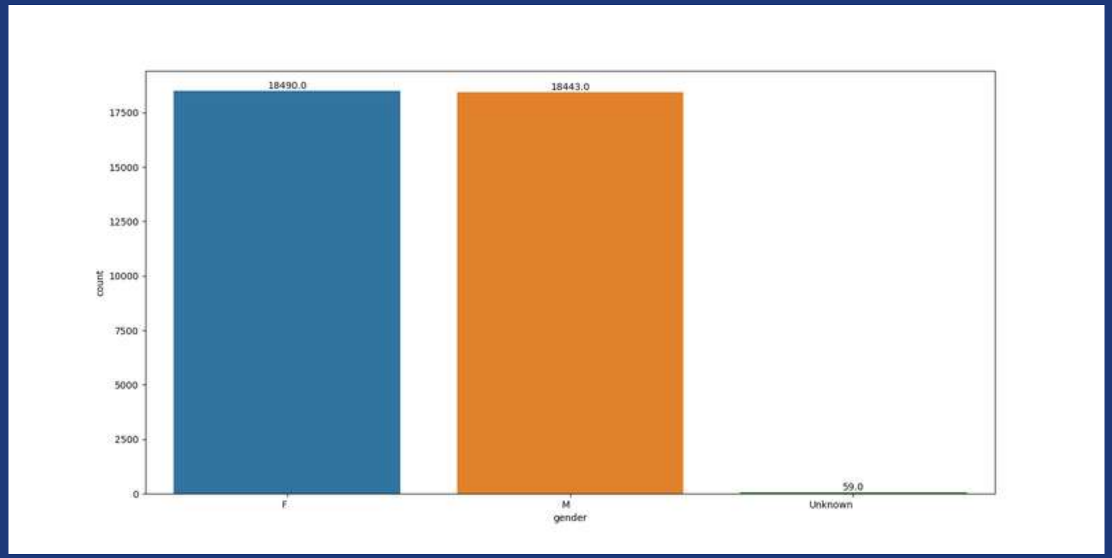




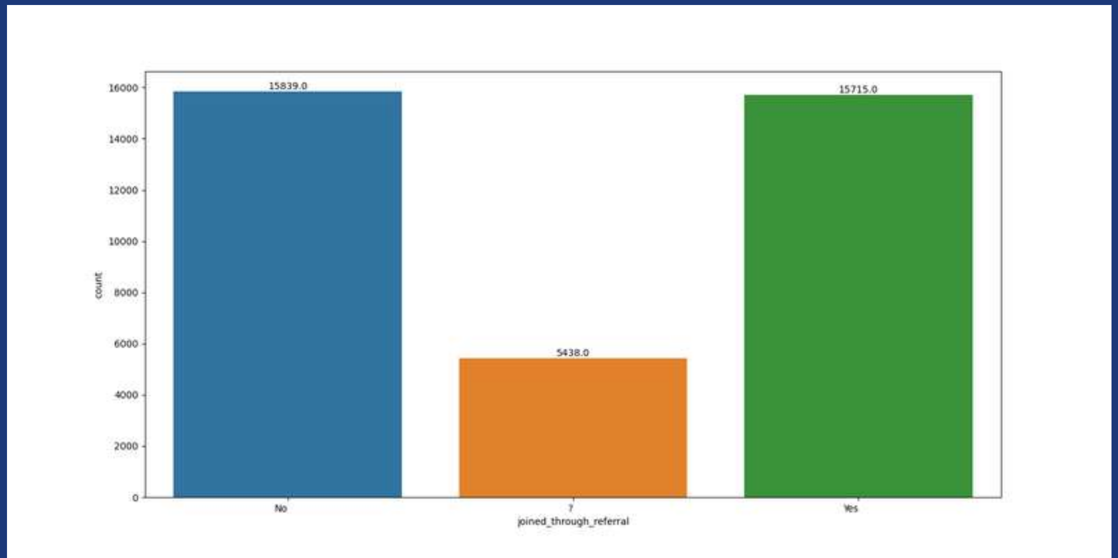
Plot diatas merupakan countplot dari variabel bertipe numerik, dari visualisasi diatas didapatkan wawasan dari value pada variabel avg_time_spent dan points_in_wallet ada yang bernilai negatif, kami berasumsi tidak ada nilai negatif pada kedua variabel tersebut sehingga kami akan membuat nilai negatif pada kedua fitur tersebut menjadi absolute (positif).

Distribusi variabel kategoris

gender



joined_through_referral

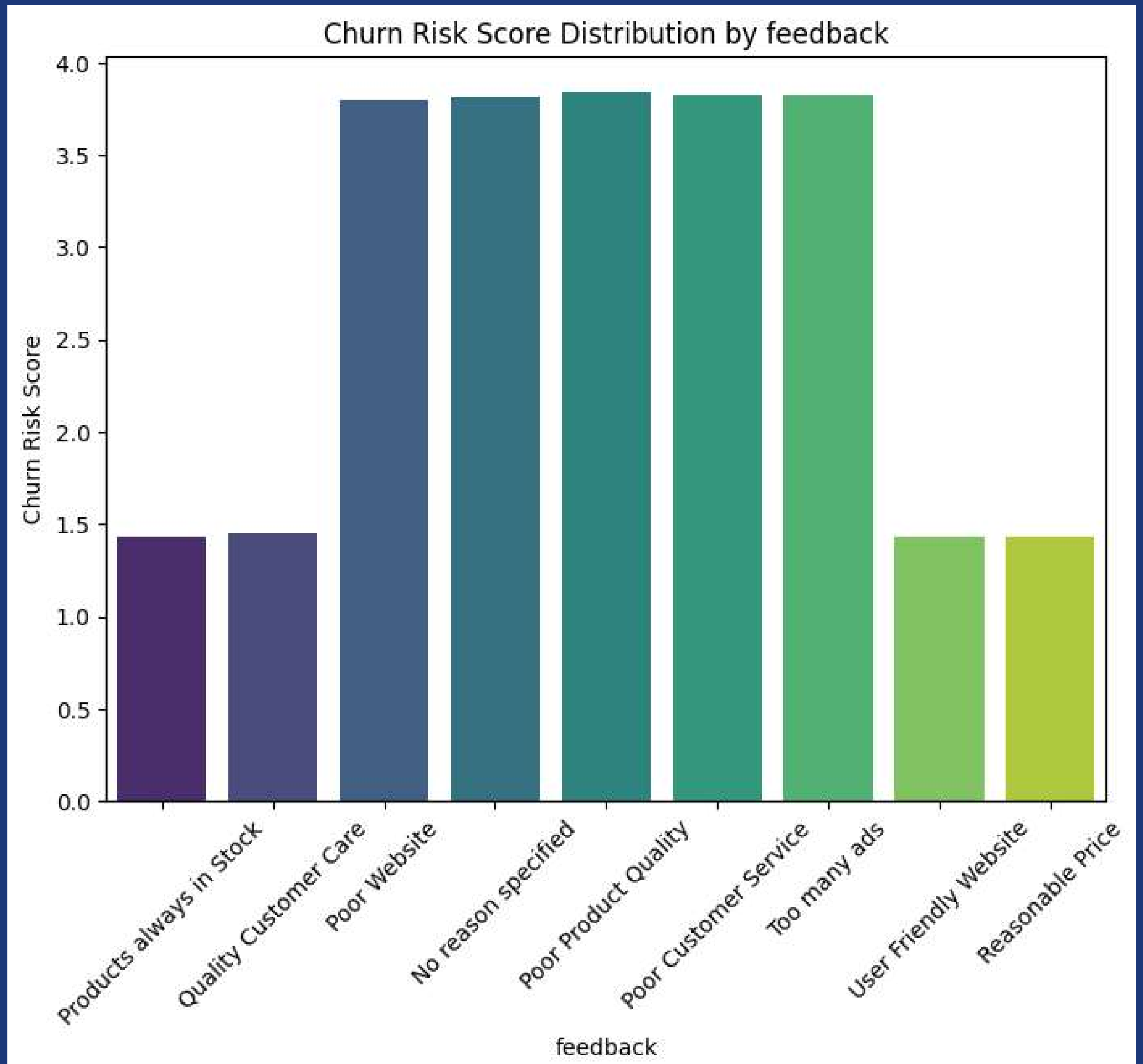


region_category

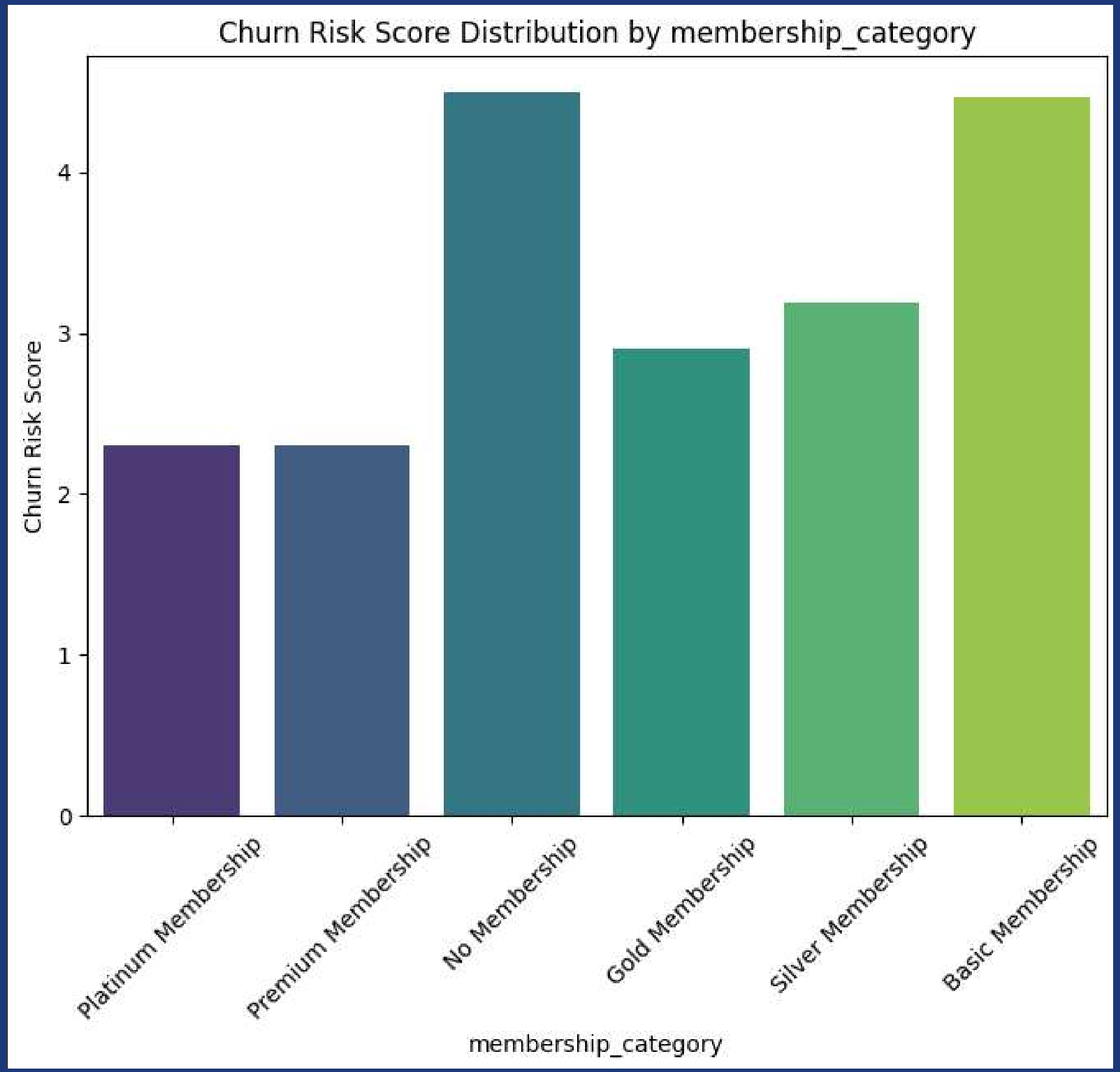
Plot diatas merupakan countplot dari variabel bertipe kategoris, dari visualisasi diatas didapatkan wawasan bahwa pada kedua fitur tersebut memiliki value "?" pada gender, "unknown" pada joined_through_referral, dan "nan" pada variabel region_category. kami mengasumsikan bahwa ketiga value tersebut merupakan sebuah missing value dalam sebuah data. Sehingga nantinya kami akan mengisi missing value tersebut dengan mengunakan modus dari data masing-masing variabel.

Distribusi variabel kategoris

Feedback

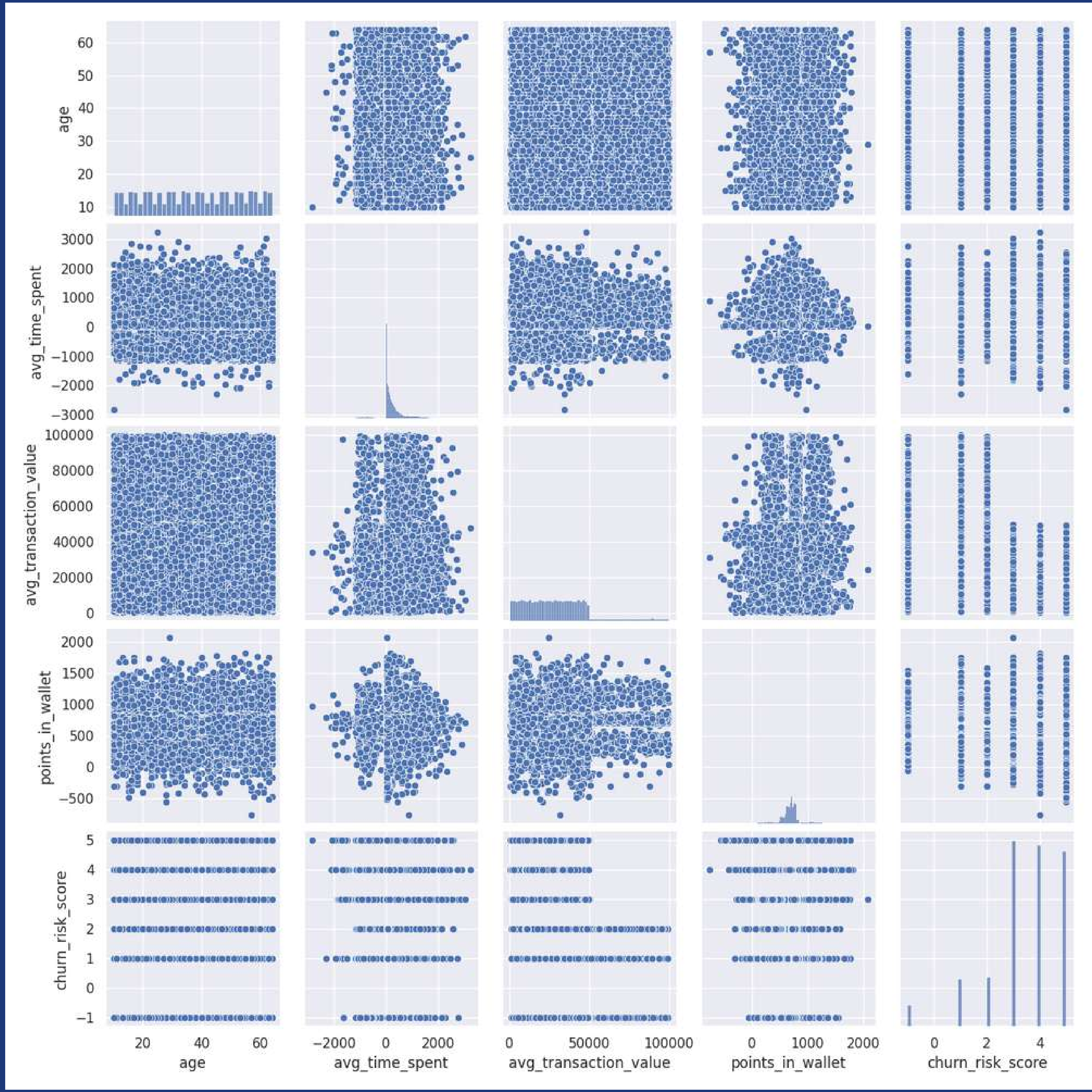


membership_category



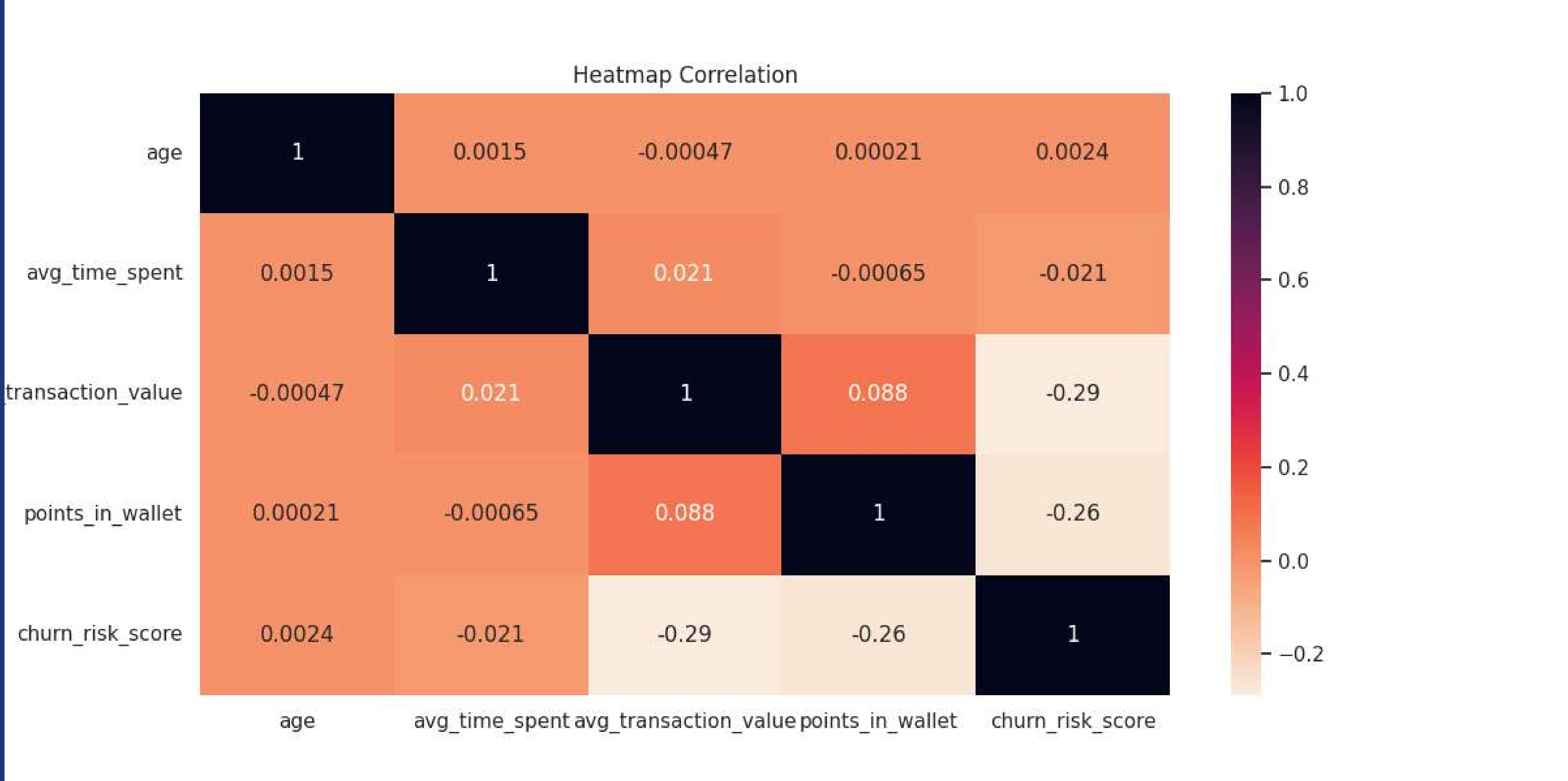
Berikut merupakan kolom categorical yang mempengaruhi churn_risk_score dapat dilihat jika feedback negatif maka churn score makin tinggi yaitu makin tinggi risiko untuk berhenti berlangganan, sedangkan untuk membership category semakin rendah tingkat membership seperti contohnya basic maka semakin tinggi pula churn risk score.

Scatter Plot



gambar disamping merupakan plot dari semua data numerikal yang disajikan dalam bentuk scatter plot untuk melihat korelasi antar 2 variabel.

Scatter Plot



Dari data heatmap diatas yang memiliki korelasi tinggi dengan variabel target (churn_risk_score) adalah variabel average transaction value yaitu -0,29. Kedua variabel tersebut memiliki korelasi negatif.

Data Pre-processing

Filling Missing Value

1. region_category (mengisi missing value dengan modus dari variabel region_category)

```
[ ] df['region_category'].isna().any()
```

```
True
```

```
[ ] def fillNan(df, col, value):
        df[col].fillna(value, inplace=True)

    # setting missing values to most occurring values
    fillNan(df, 'region_category', df['region_category'].mode()[0])
```

```
[ ] df['region_category'].isna().any()
```

```
False
```

2. points_in_wallet (mengisi missing value dengan rata-rata data)

```
[ ] df['points_in_wallet'].isna().any()
```

```
True
```

```
# setting missing values to most occurring values
fillNan(df, 'points_in_wallet', df['points_in_wallet'].mean())
df['points_in_wallet'].isna().any()
```

```
False
```

Filling Missing Value

3. joined_through_referral (mengganti value "?" dengan "No")

```
# setting missing values to most occurring values
df['joined_through_referral'].unique()
```

```
array(['No', '?', 'Yes'], dtype=object)
```

```
def replace_question_mark(value):
    if value == '?':
        return 'No'
    return value

df['joined_through_referral'] = df['joined_through_referral'].apply(replace_question_mark)

print(df['joined_through_referral'].unique())
```

```
['No' 'Yes']
```

4. avg_frequency_login_days (mengganti nilai "Error" dengan "0")

```
df['avg_frequency_login_days']
```

```
0         17.0
1         18.0
2         22.0
3          6.0
4         16.0
          ...
36987      6.0
36988     28.0
36989    Error
36990     20.0
36991    Error
Name: avg_frequency_login_days, Length: 36992, dtype: object
```

```
# setting missing values to most occurring values
df['avg_frequency_login_days'] = df['avg_frequency_login_days'].apply(lambda x:0 if x == 'Error' else x)
df['avg_frequency_login_days'].describe()
```

```
count     36992
unique     1654
top           0
freq       3522
Name: avg_frequency_login_days, dtype: int64
```

Value Replacement

```
#mengubah nilai churn rate -1 jadi 1
df['churn_risk_score'] = df['churn_risk_score'].apply(lambda x:1 if x == -1 else x)
df['churn_risk_score'].unique()
```

```
array([2, 1, 5, 3, 4], dtype=int64)
```

Pada kolom target terdapat data -1 yang akan diubah menjadi 1 sehingga tidak terdapat nilai "-" (minus), karena churn risk score -1 memiliki arti risiko untuk berhenti berlangganan (churn) rendah sehingga disatukan/digolongkan bersama nilai 1

```
# Mengganti nilai "unknown" dalam kolom "gender" dengan modus
mode_gender = df['gender'].mode()[0]
df['gender'] = df['gender'].replace('Unknown', mode_gender)
df['gender'].unique()
```

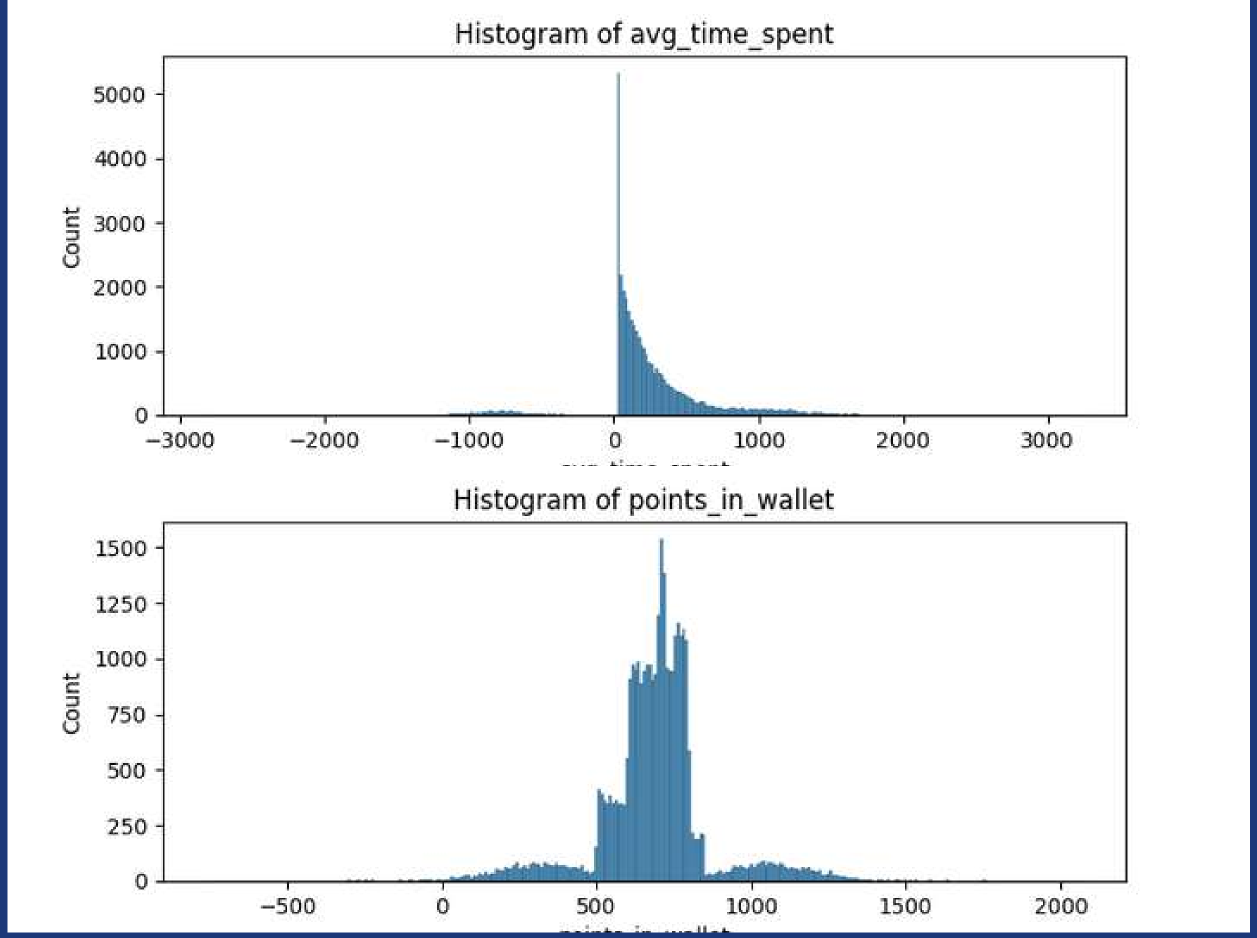
```
array(['F', 'M'], dtype=object)
```

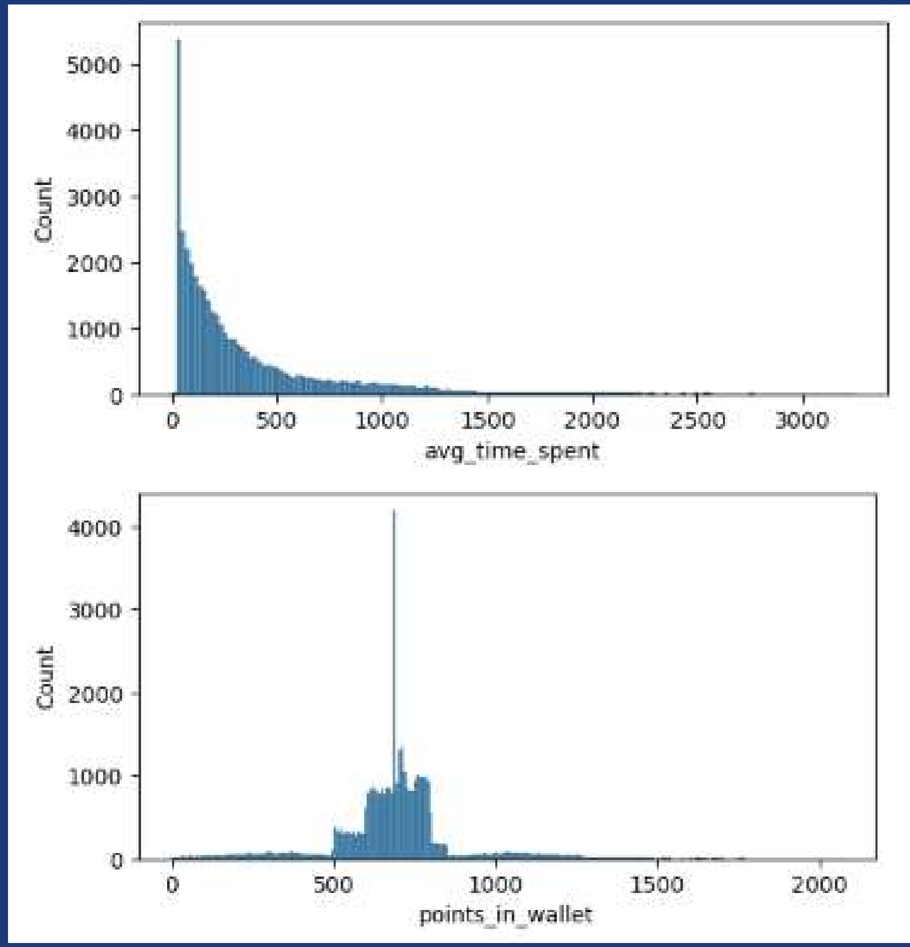
Mengubah value "Unknown" pada variabel gender dengan modus data

Value Replacement

```
[ ] # Membuat semua nilai negatif menjadi positif pada kolom avg_time_spent dan point_in_wallet
    df['avg_time_spent'] = df['avg_time_spent'].apply(lambda x: abs(x))
    df['points_in_wallet'] = df['points_in_wallet'].apply(lambda x: abs(x))
```

Membuat kolom avg_time_spent dan point in wallet menjadi positif (asumsi jika - adalah kesalahan) karena seharusnya nilai time spent dan point in wallet tidak negatif





Feature Transforming

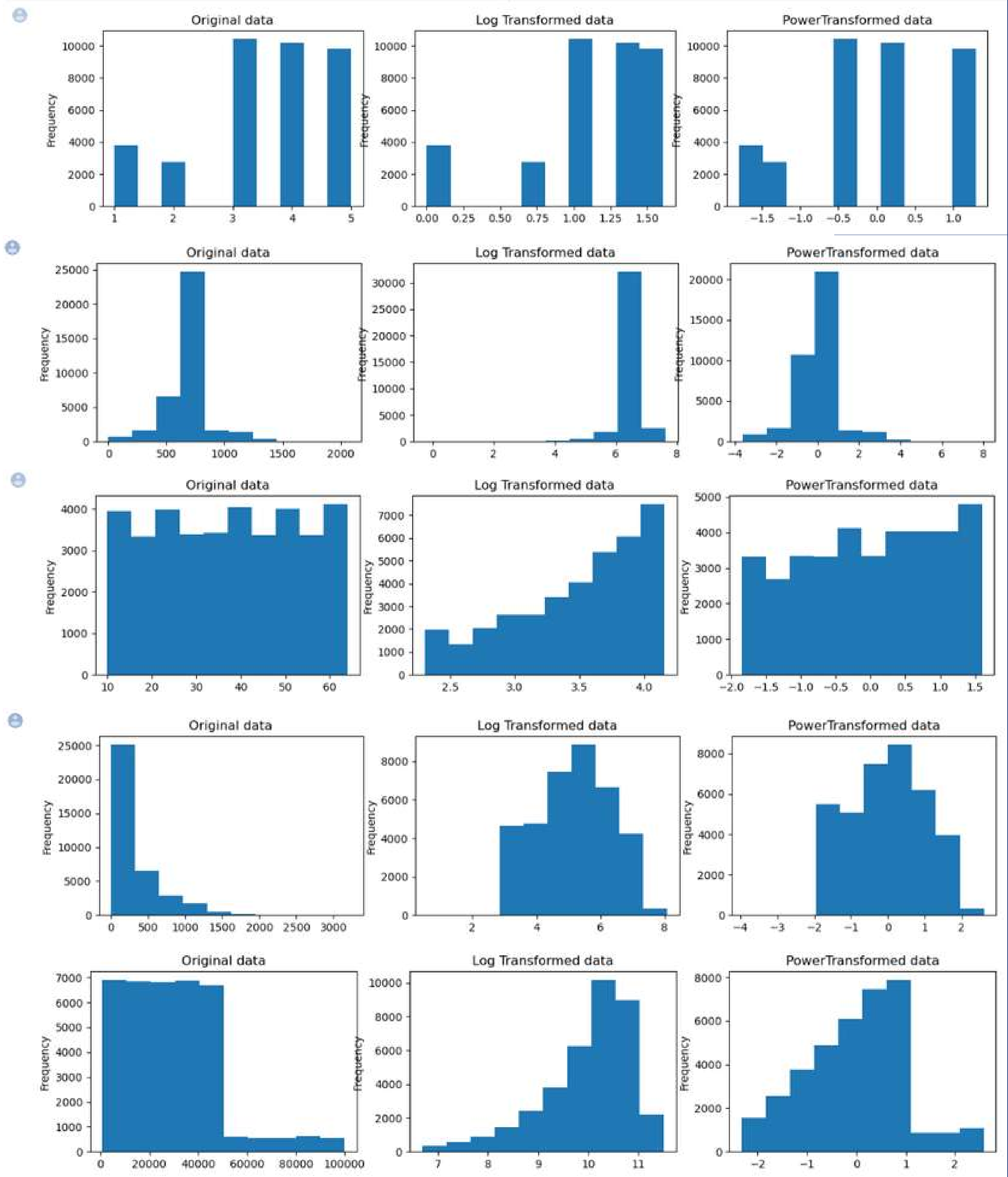
H0: Distribusi Normal
H1: Distribusi tidak normal

```
from scipy import stats

normaltest_result_churn = stats.normaltest(df['churn_risk_score'])[1]
normaltest_result_age = stats.normaltest(df['age'])[1]
normaltest_result_points_in_wallet = stats.normaltest(df['points_in_wallet'])[1]
normaltest_result_avg_time_spent = stats.normaltest(df['avg_time_spent'])[1]
normaltest_result_avg_transaction_value = stats.normaltest(df['avg_transaction_value'])[1]
```

```
The p-value for the null hypothesis of the Churn Risk Score not being Normally distributed is 0.0
The p-value for the null hypothesis of the Age not being Normally distributed is 0.0
The p-value for the null hypothesis of the Age not being Normally distributed is 0.0
The p-value for the null hypothesis of the Age not being Normally distributed is 0.0
The p-value for the null hypothesis of the Age not being Normally distributed is 0.0
```

Karena p-value kurang dari 0.05 maka H0 ditolak atau distribusi tidak normal. Untuk menormalisasi data dilakukan log transform dan power transform.

Distribusi Normal



churn risk score

Points in wallet

Age

Avg time spent

Avg transaction value

```
[ ] df['transf_PIW'] = log_transformed_PIW
    df['transf_ATS'] = log_transformed_ATS
    df['transf_ATV'] = log_transformed_ATV

[ ] df_transformed = df.drop(['points_in_wallet', 'avg_time_spent', 'avg_transaction_value'], axis=1)
    df=df_transformed
```

Setelah dilakukan power transform dan log transform pada ke lima fitur tersebut. variabel yang mengalami perubahan yang signifikan setelah dilakukan log transform hanyalah points_in_wallet, avg_time_spent, dan avg_transaction value. sehingga hanya ketiga variabel ini saja yang data log transform nya akan dipakai.

Feature Engineering I

Encoding I

ONE HOT ENCODING

```
[ ] # extract numerical and categorical for dummy and scaling later
    custom_feat = ['region_category', 'complaint_status', 'feedback']
    encode_data = df.copy()  # Duplikasi data df untuk operasi one-hot encoding

    for feat in cat_features.columns:
        if len(df[feat].unique()) > 2 and feat in custom_feat:
            dummyVars = pd.get_dummies(encode_data[feat], drop_first=True, prefix=feat+"_")
            encode_data = pd.concat([encode_data, dummyVars], axis=1)
            encode_data.drop(feat, axis=1, inplace=True)

    encode_data
```

LABEL ENCODING

```
cols = ['membership_category', 'gender',
        'joined_through_referral', 'used_special_discount',
        'offer_application_preference', 'past_complaint']

encoders = {}

for c in cols:
    lbl = LabelEncoder()
    lbl.fit(list(encode_data[c].values))
    encode_data[c] = lbl.transform(list(encode_data[c].values))
    encoders[c] = lbl
```

```
encode_data
```

	age	gender	membership_category	joined_through_referral	avg_frequency_login_days	used_special_discount	offer_application_preference	past_complaint	churn_risk_score	transf_PIW	...
0	18	0	3	0	17.0	1	1	0	2	6.661535	
1	32	0	4	0	10.0	1	0	1	1	6.532163	
2	44	0	2	1	22.0	0	1	1	5	6.215987	
3	37	1	2	1	6.0	0	1	1	5	6.341523	
4	31	0	2	0	16.0	0	1	1	5	6.496865	
...	...	...	...	...	...	...	...	...	...	...	
36987	46	0	0	0	6.0	0	1	1	4	6.460702	
36988	29	0	0	0	28.0	1	0	0	5	6.269077	
36989	23	0	0	1	0.0	0	1	1	4	6.522784	
36990	53	1	3	0	20.0	1	1	0	3	5.284545	
36991	35	1	5	0	0.0	1	0	0	2	6.579210	

36992 rows × 26 columns

Split Data

```
[ ]  response = encode_data['churn_risk_score']

     encode_data = encode_data.drop(columns='churn_risk_score')
```

```
[ ]  from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(encode_data, response,
                                                         stratify=response,
                                                         test_size = 0.2, #use 0.1 if data is huge.
                                                         random_state = 0)

     #to resolve any class imbalance - use stratify parameter.

     print("Number transactions X_train encode_data: ", X_train.shape)
     print("Number transactions y_train encode_data: ", y_train.shape)
     print("Number transactions X_test encode_data: ", X_test.shape)
     print("Number transactions y_test encode_data: ", y_test.shape)
```

```
Number transactions X_train encode_data:  (29593, 25)
Number transactions y_train encode_data:  (29593,)
Number transactions X_test encode_data:  (7399, 25)
Number transactions y_test encode_data:  (7399,)
```

Data Train 80%
Data Test 20%

Data Scalling (StandardScaler)

```
[ ] from sklearn.preprocessing import StandardScaler

    sc_X = StandardScaler()
    X_train2 = pd.DataFrame(sc_X.fit_transform(X_train))
    X_train2.columns = X_train.columns.values
    X_train2.index = X_train.index.values
    X_train = X_train2

    X_test2 = pd.DataFrame(sc_X.transform(X_test))
    X_test2.columns = X_test.columns.values
    X_test2.index = X_test.index.values
    X_test = X_test2
```

	age	gender	membership_category	joined_through_referral	avg_frequency_login_days	used_special_discount	offer_application_preference	past_complaint	transf_PIW	transf_ATS	...
34522	0.248893	1.006339	1.014573	-0.858300	-0.344429	-1.108781	0.903062	1.004369	0.343640	1.682524	
11435	-0.066020	1.006339	0.437805	1.165094	0.157836	-1.108781	0.903062	1.004369	0.293789	1.248898	
7050	-1.703565	-0.993701	0.437805	-0.858300	0.157836	0.901891	-1.107343	1.004369	1.013414	-0.058454	
18211	0.311875	-0.993701	1.014573	1.165094	0.760553	0.901891	-1.107343	-0.995650	0.519498	1.242043	
27687	-0.884792	1.006339	0.437805	-0.858300	-0.645788	0.901891	-1.107343	1.004369	-1.935814	-0.858981	
...	...	...	...	...	...	...	...	...	...	...	
32610	0.878718	-0.993701	1.014573	-0.858300	1.262818	0.901891	-1.107343	-0.995650	0.526709	-1.592956	
24030	0.689771	1.006339	-0.715733	-0.858300	0.258289	0.901891	-1.107343	1.004369	-0.054765	-0.247746	
17960	-1.262687	-0.993701	-0.138964	-0.858300	0.057383	0.901891	-1.107343	-0.995650	-0.340095	-1.209687	
35931	-1.073740	1.006339	-1.292501	-0.858300	1.162365	0.901891	-1.107343	1.004369	0.125465	0.006921	
34063	-0.003037	-0.993701	-0.715733	1.165094	0.660100	-1.108781	0.903062	1.004369	-0.224484	-0.746051	

29593 rows × 25 columns

Feature Engineering 2

Encoding 2

Label Encoding

```
[ ]  cols = cat_features
     encoders = {}

     for c in cols:
         lbl = LabelEncoder()
         lbl.fit(list(encode_data_2[c].values))
         encode_data_2[c] = lbl.transform(list(encode_data_2[c].values))
         encoders[c] = lblcols = cat_features
     encoders = {}
```

```
[ ]  encode_data_2
```

	age	gender	region_category	membership_category	joined_through_referral	avg_frequency_login_days	used_special_discount	offer_application_preference	past_complaint	complaint_status	feedback	cl
0	18	0	2	3	0	17.0	1	1	0	1	4	
1	32	0	0	4	0	10.0	1	0	1	2	5	
2	44	0	1	2	1	22.0	0	1	1	3	3	
3	37	1	0	2	1	6.0	0	1	1	4	3	
4	31	0	0	2	0	16.0	0	1	1	2	3	
...	...	...	...	...	...	...	...	...	...	...	...	
36987	46	0	1	0	0	6.0	0	1	1	0	0	
36988	29	0	1	0	0	28.0	1	0	0	1	1	
36989	23	0	1	0	1	0.0	0	1	1	4	3	
36990	53	1	2	3	0	20.0	1	1	0	1	0	
36991	35	1	1	5	0	0.0	1	0	0	1	5	

36992 rows × 15 columns

Split Data

```
[ ] response_2 = encode_data_2['churn_risk_score']

    encode_data_2 = encode_data_2.drop(columns='churn_risk_score')
```

```
[ ] from sklearn.model_selection import train_test_split

    X_train_2, X_test_2, y_train_2, y_test_2 = train_test_split(encode_data_2, response_2,
                                                    stratify=response_2,
                                                    test_size = 0.2, #use 0.1 if data is huge.
                                                    random_state = 0)

    #to resolve any class imbalance - use stratify parameter.

    print("Number transactions X_train_2 encode_data_2: ", X_train_2.shape)
    print("Number transactions y_train_2 encode_data_2: ", y_train_2.shape)
    print("Number transactions X_test_2 encode_data_2: ", X_test_2.shape)
    print("Number transactions y_test_2 encode_data_2: ", y_test_2.shape)
```

```
Number transactions X_train_2 encode_data_2:  (29593, 14)
Number transactions y_train_2 encode_data_2:  (29593,)
Number transactions X_test_2 encode_data_2:  (7399, 14)
Number transactions y_test_2 encode_data_2:  (7399,)
```

Data Train 80%
Data Test 20%

Data Scalling (StandardScaler)

```
[ ] from sklearn.preprocessing import StandardScaler

    sc_X2 = StandardScaler()
    X_train2_2 = pd.DataFrame(sc_X2.fit_transform(X_train_2))
    X_train2_2.columns = X_train_2.columns.values
    X_train2_2.index = X_train_2.index.values
    X_train_2 = X_train2_2

    X_test2_2 = pd.DataFrame(sc_X2.transform(X_test_2))
    X_test2_2.columns = X_test_2.columns.values
    X_test2_2.index = X_test_2.index.values
    X_test_2 = X_test2_2
```

	age	gender	region_category	membership_category	joined_through_referral	avg_frequency_login_days	used_special_discount	offer_application_preference	past_complaint	complaint_status
34522	0.248893	1.006339	1.871055	1.014573	-0.858300	-0.344429	-1.108781	0.903062	1.004369	0.305973
11435	-0.066020	1.006339	1.871055	0.437805	1.165094	0.157836	-1.108781	0.903062	1.004369	1.126291
7050	-1.703565	-0.993701	0.334404	0.437805	-0.858300	0.157836	0.901891	-1.107343	1.004369	-1.334663
18211	0.311875	-0.993701	0.334404	1.014573	1.165094	0.760553	0.901891	-1.107343	-0.995650	-0.514345
27687	-0.884792	1.006339	0.334404	0.437805	-0.858300	-0.645788	0.901891	-1.107343	1.004369	0.305973
...	...	...	...	...	...	...	...	...	...	...
32610	0.878718	-0.993701	-1.202246	1.014573	-0.858300	1.262818	0.901891	-1.107343	-0.995650	-0.514345
24030	0.689771	1.006339	0.334404	-0.715733	-0.858300	0.258289	0.901891	-1.107343	1.004369	1.946609
17960	-1.262687	-0.993701	0.334404	-0.138964	-0.858300	0.057383	0.901891	-1.107343	-0.995650	-0.514345
35931	-1.073740	1.006339	-1.202246	-1.292501	-0.858300	1.162365	0.901891	-1.107343	1.004369	1.126291
34063	-0.003037	-0.993701	-1.202246	-0.715733	1.165094	0.660100	-1.108781	0.903062	1.004369	1.946609

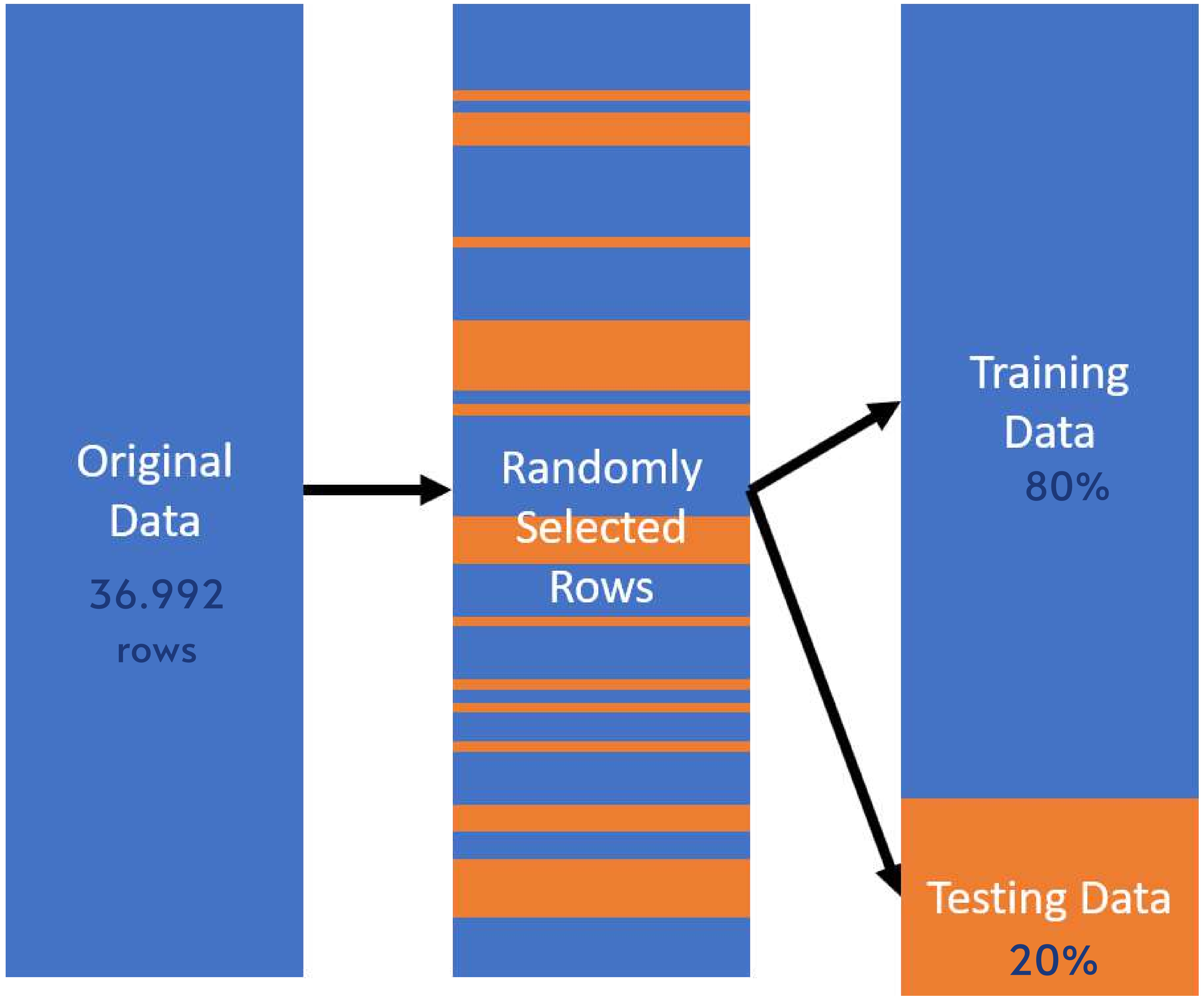
29593 rows × 14 columns

Modelling

Modelling Step

1. Split the data
2. Modelling the data with a few machine learning models using 2 types of encode
3. Models evaluation
4. Compare the models
5. Choose the best machine learning model
6. Optimize the model with hyperparameter tuning
7. Get the best machine learning model

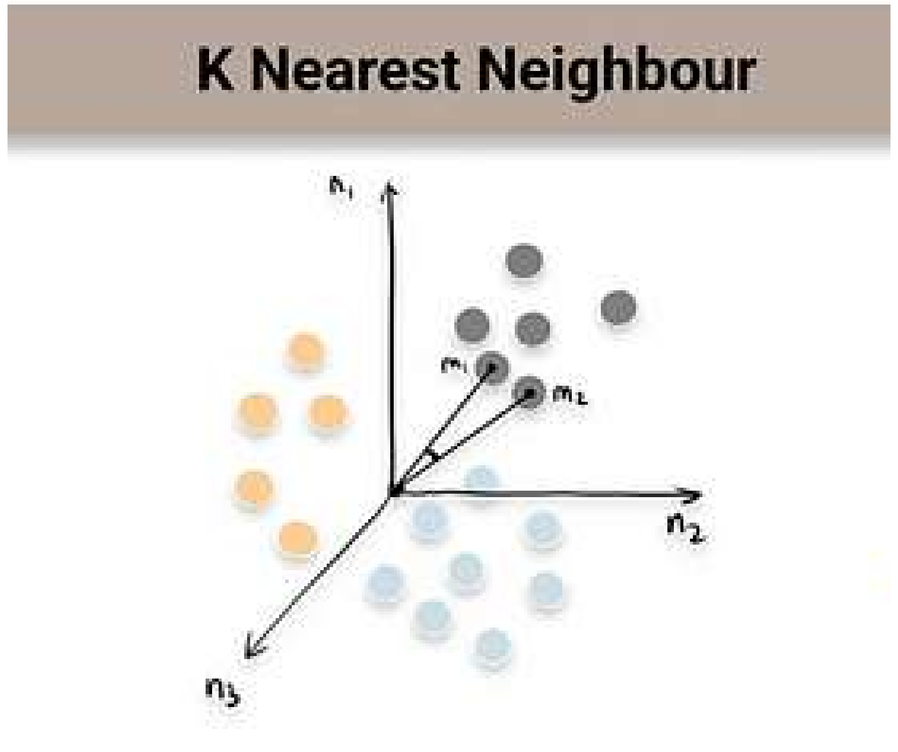
Data Split

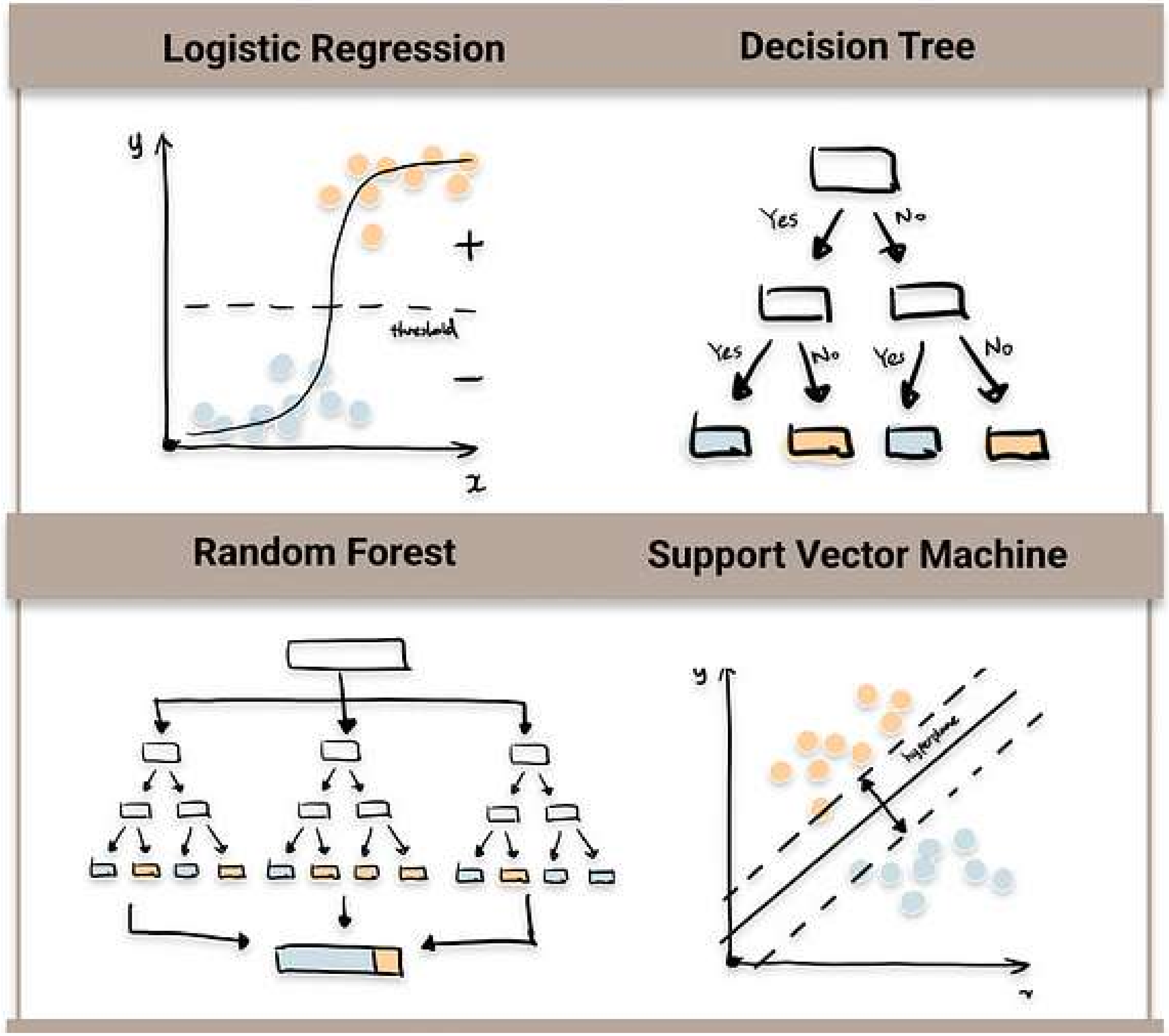


Original Data
36.992 rows
Randomly Selected Rows
Training Data
80%
Testing Data
20%

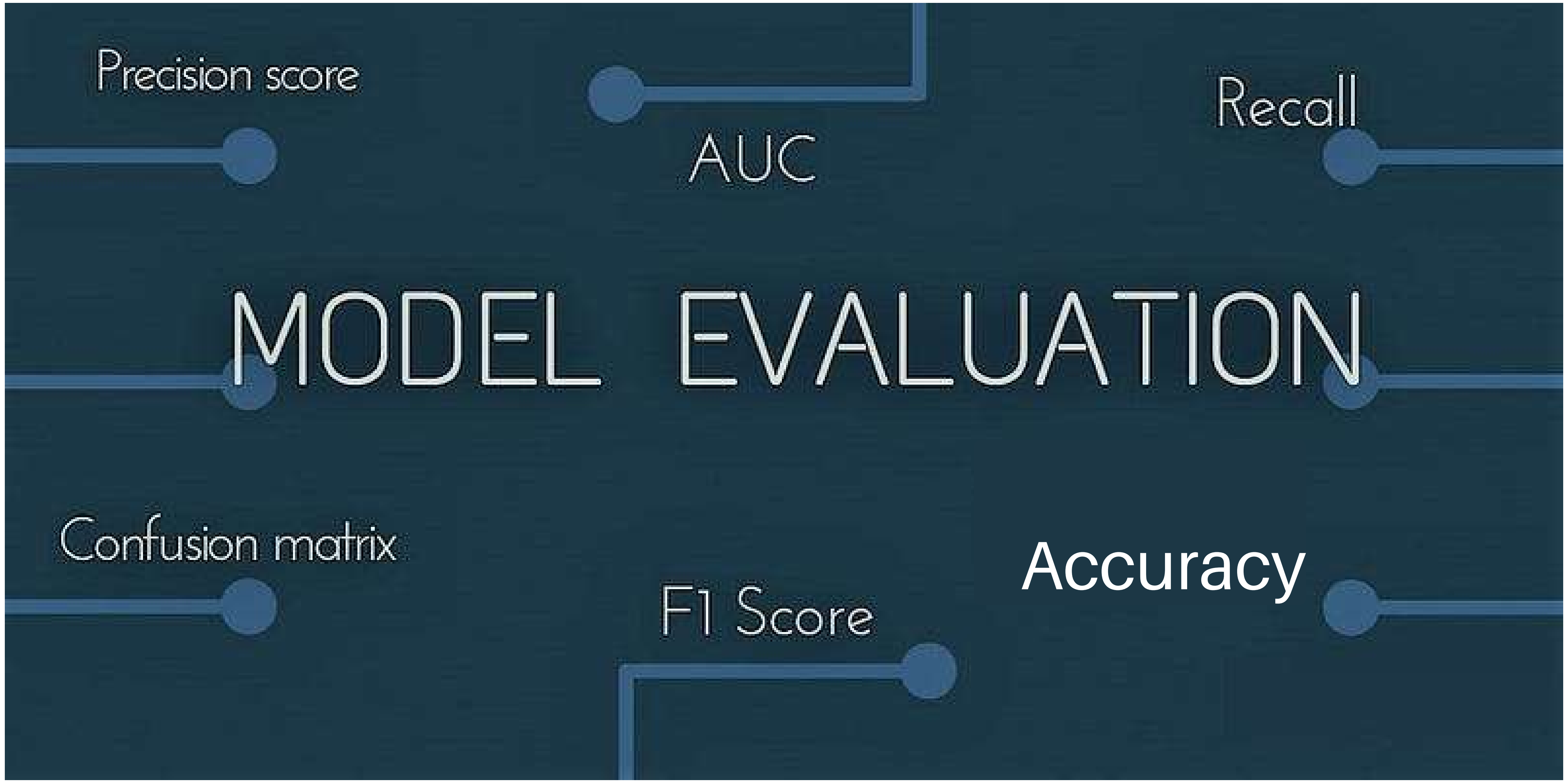
Modelling

Model for classification





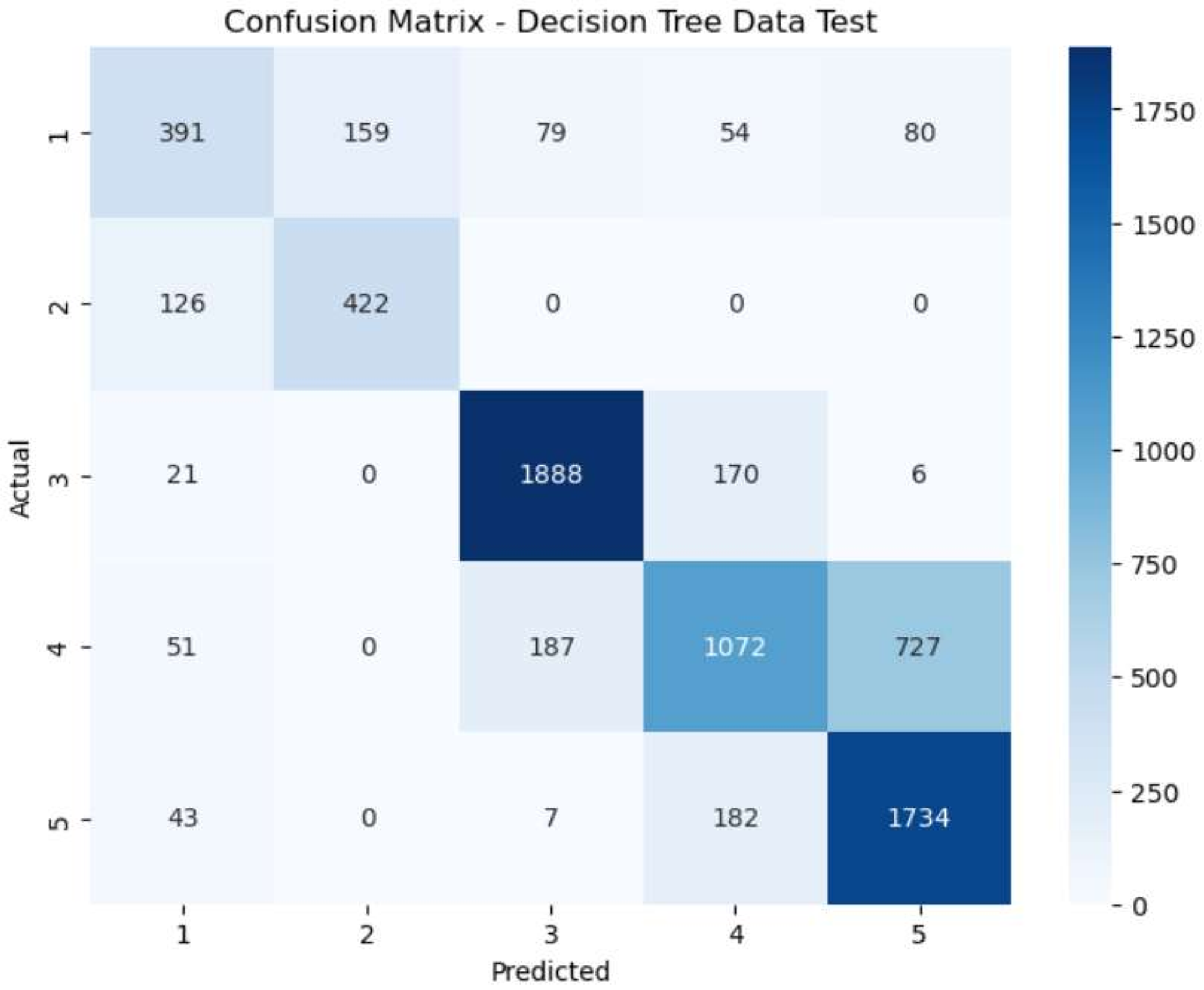
Model Evaluation

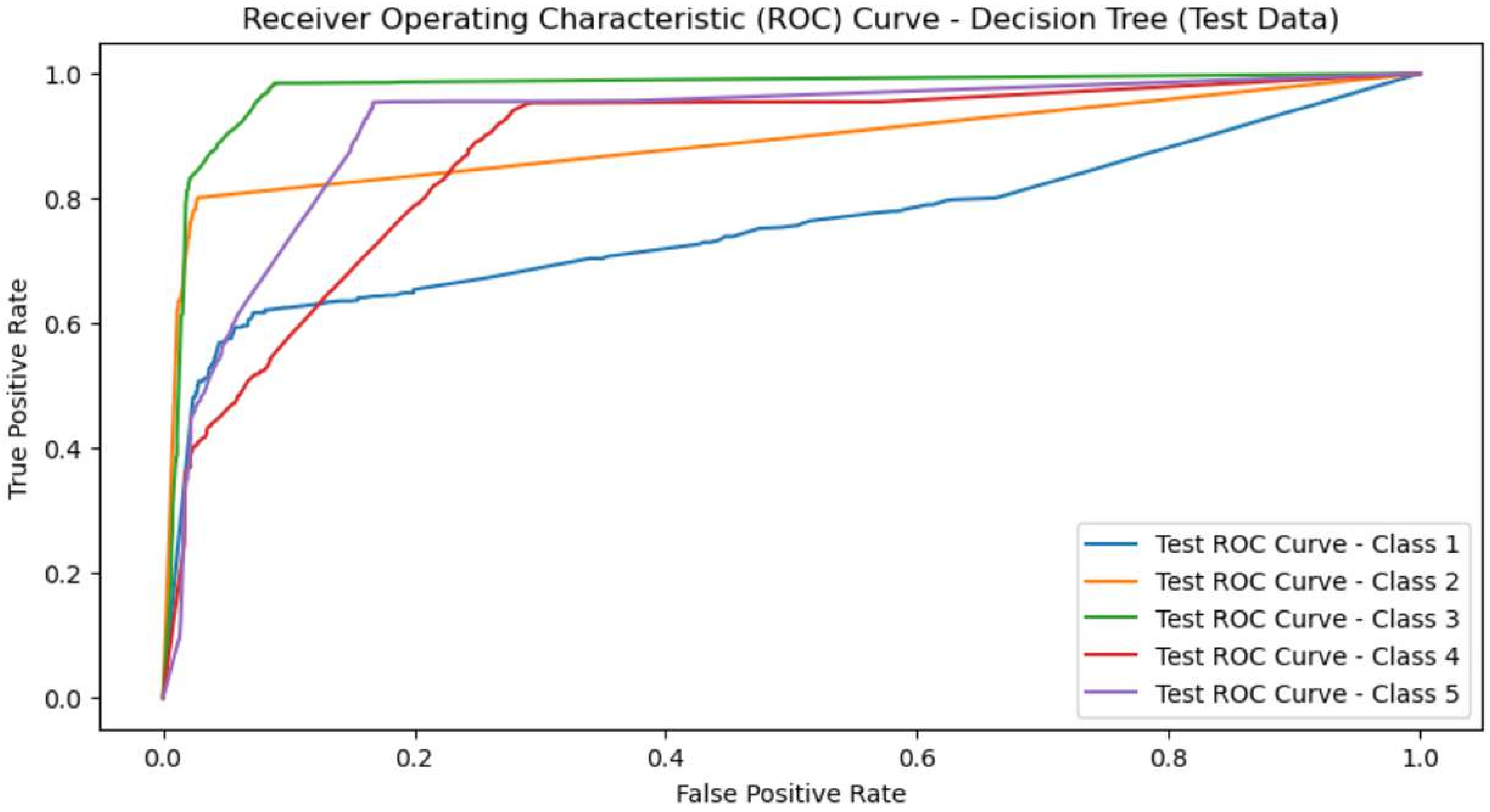


Model with One Hot + Label Encoder

Decision Tree

	Accuracy	Precision	Recall	F1 Score	F2 Score
Data Train	0.8347	0.8581	0.8431	0.8414	0.8403
Data Test	0.7443	0.7250	0.7193	0.7152	0.7160

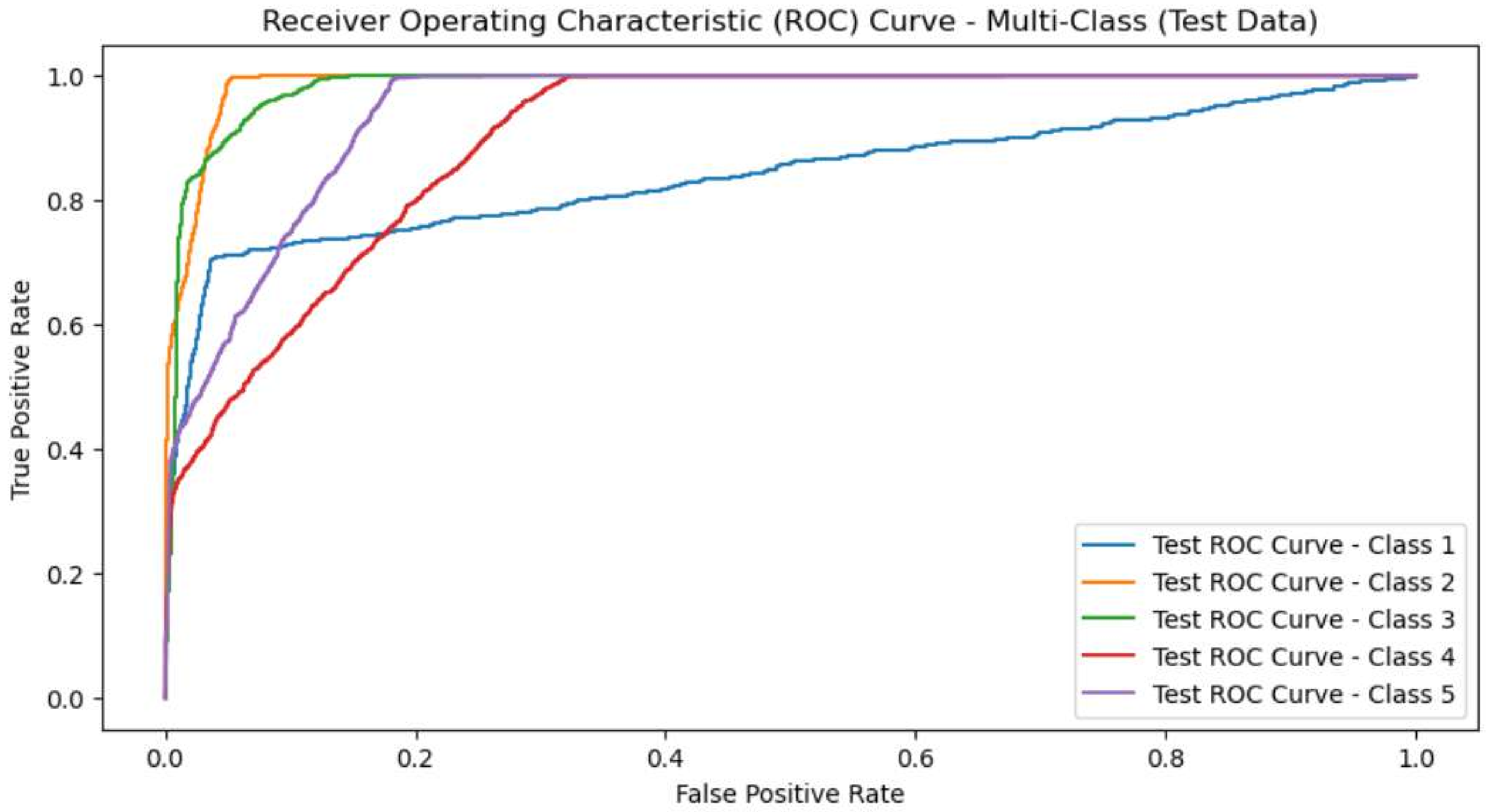




Model with One Hot + Label Encoder

Random Forest

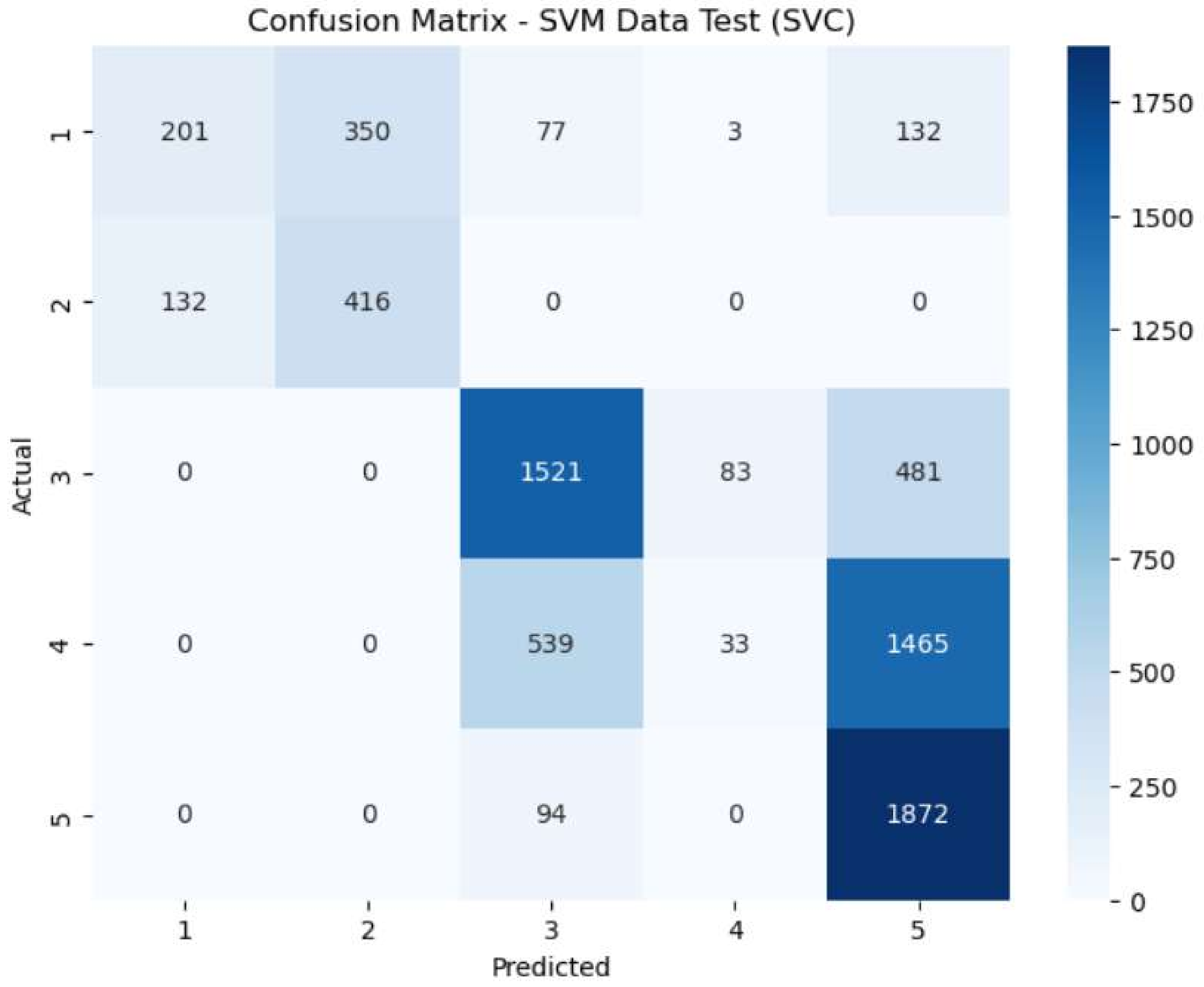
	Accuracy	Precision	Recall	F1 Score	F2 Score
Data Train	0.8950	0.9255	0.8915	0.9019	0.8942
Data Test	0.7551	0.7515	0.7266	0.7290	0.7254

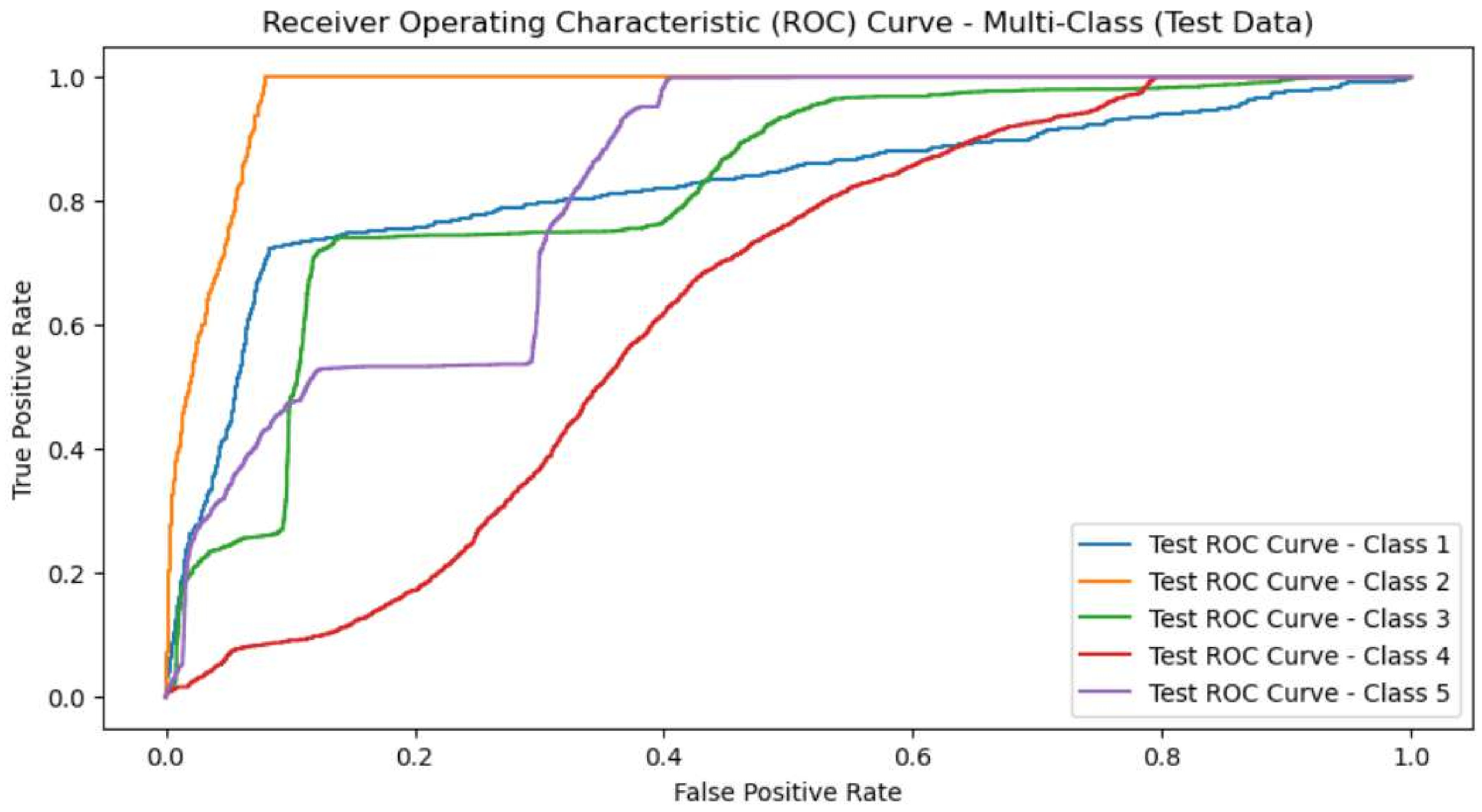


Model with One Hot + Label Encoder

SVM

	Accuracy	Precision	Recall	F1 Score	F2 Score
SVM Train (SVC)	0.5437	0.5056	0.5412	0.4670	0.5014
SVM Test (SVC)	0.5464	0.5159	0.5441	0.4737	0.5063

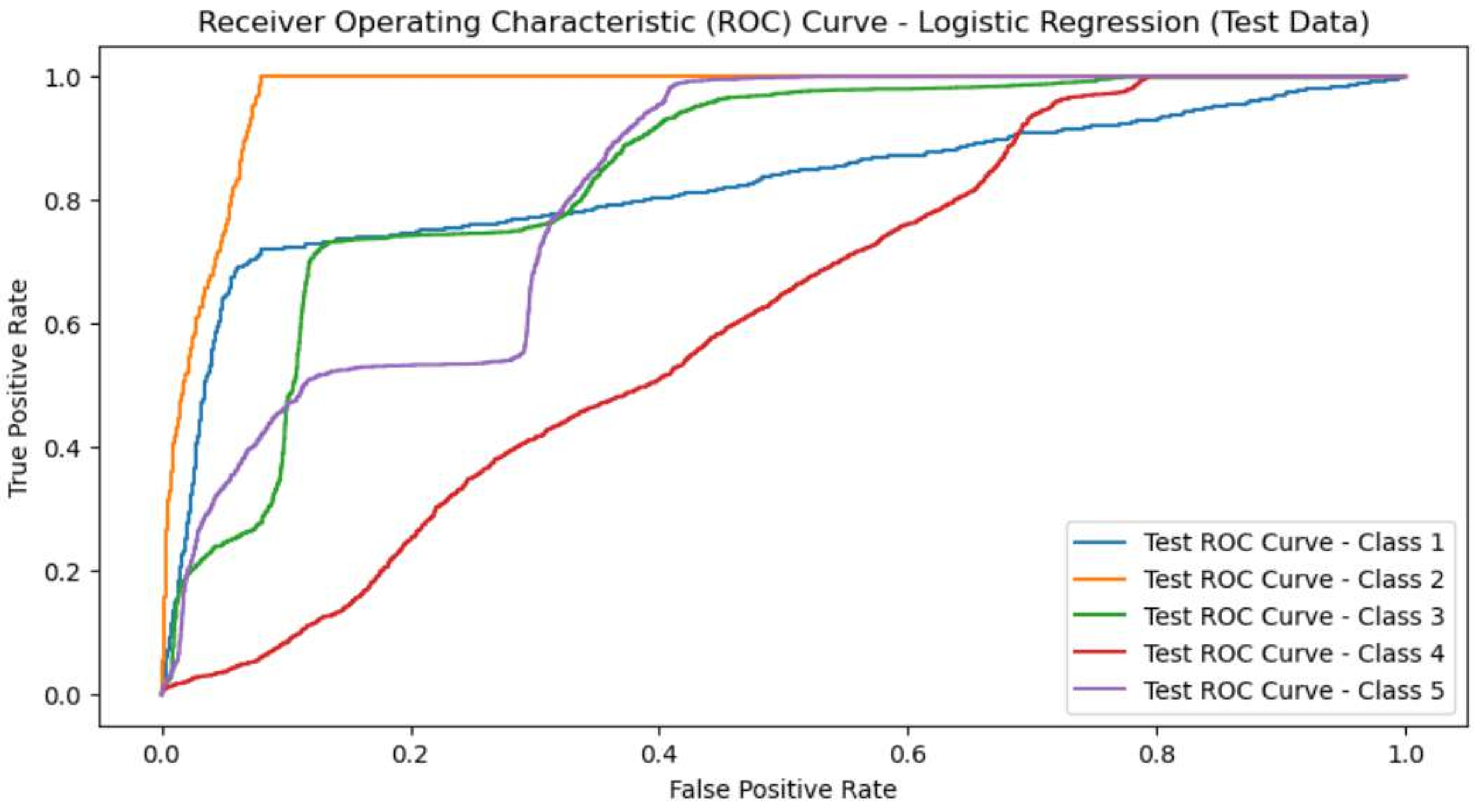




Model with One Hot + Label Encoder

Logistic Regression

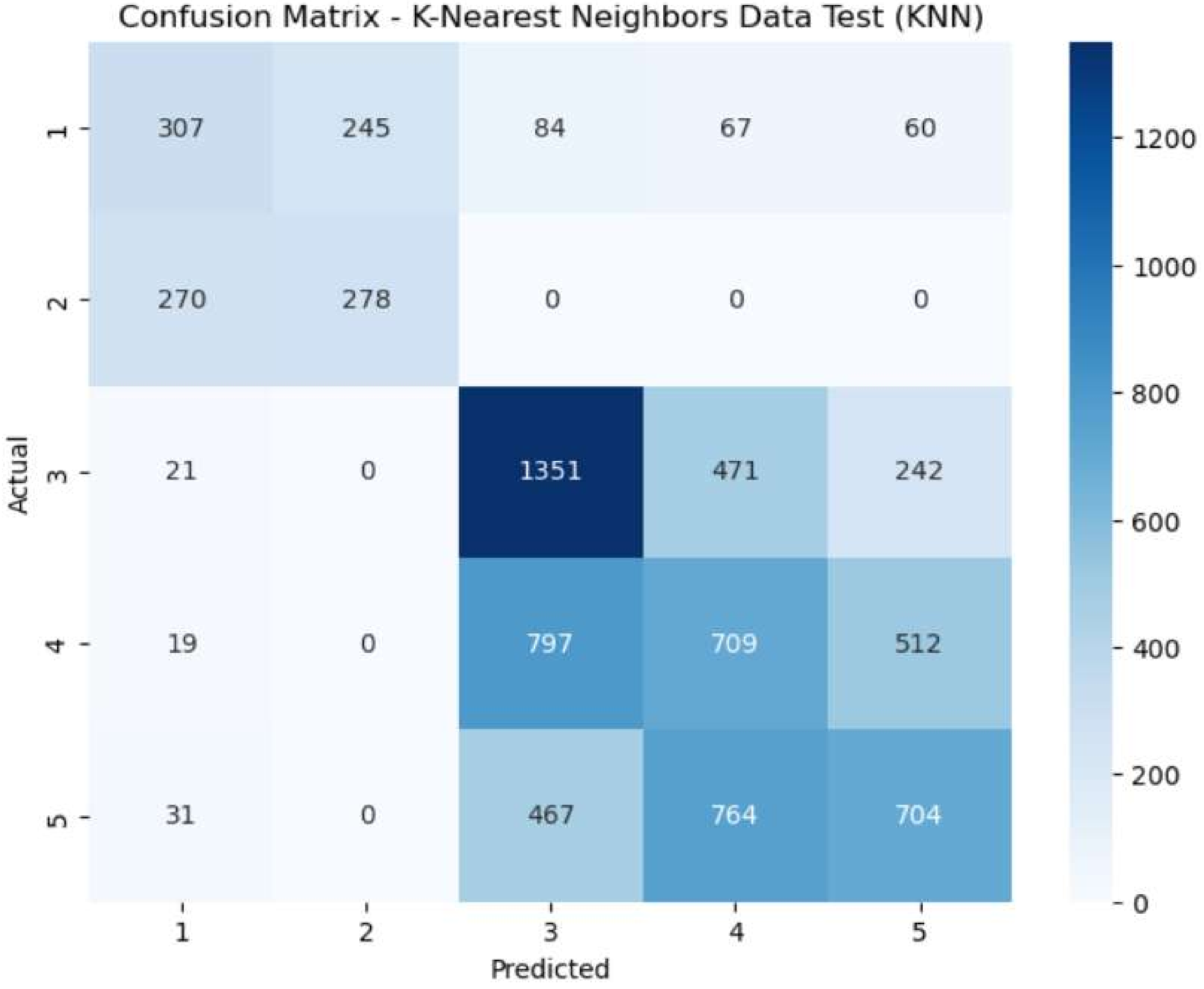
	Accuracy	Precision	Recall	F1 Score	F2 Score
Logistic Regression Train	0.4893	0.5041	0.4983	0.4782	0.4863
Logistic Regression Test	0.4947	0.5095	0.5026	0.4851	0.4920

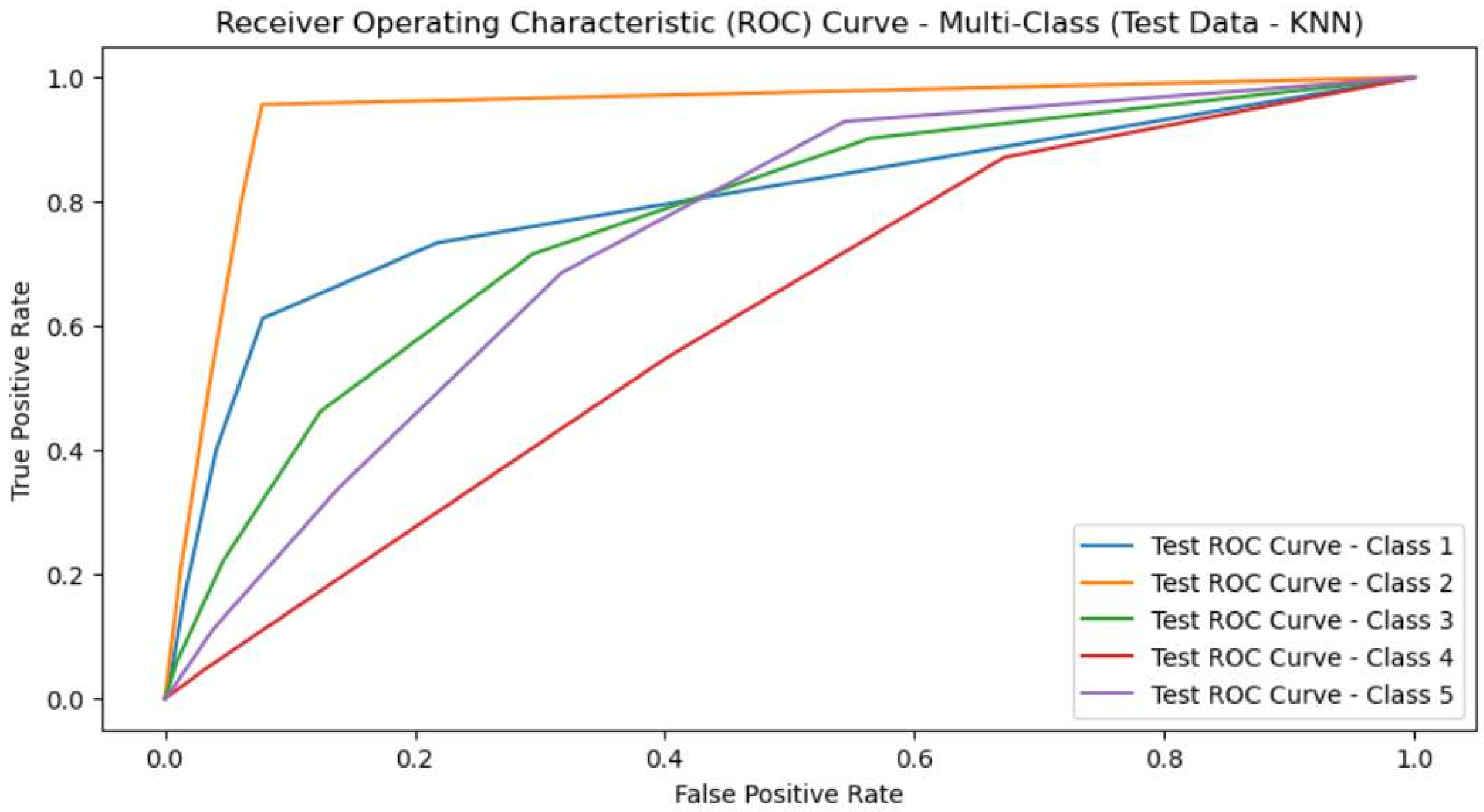


Model with One Hot + Label Encoder

KNN

	Accuracy	Precision	Recall	F1 Score	F2 Score
KNN Train	0.6402	0.6571	0.6356	0.6425	0.6374
KNN Test	0.4526	0.4644	0.4528	0.4547	0.4526

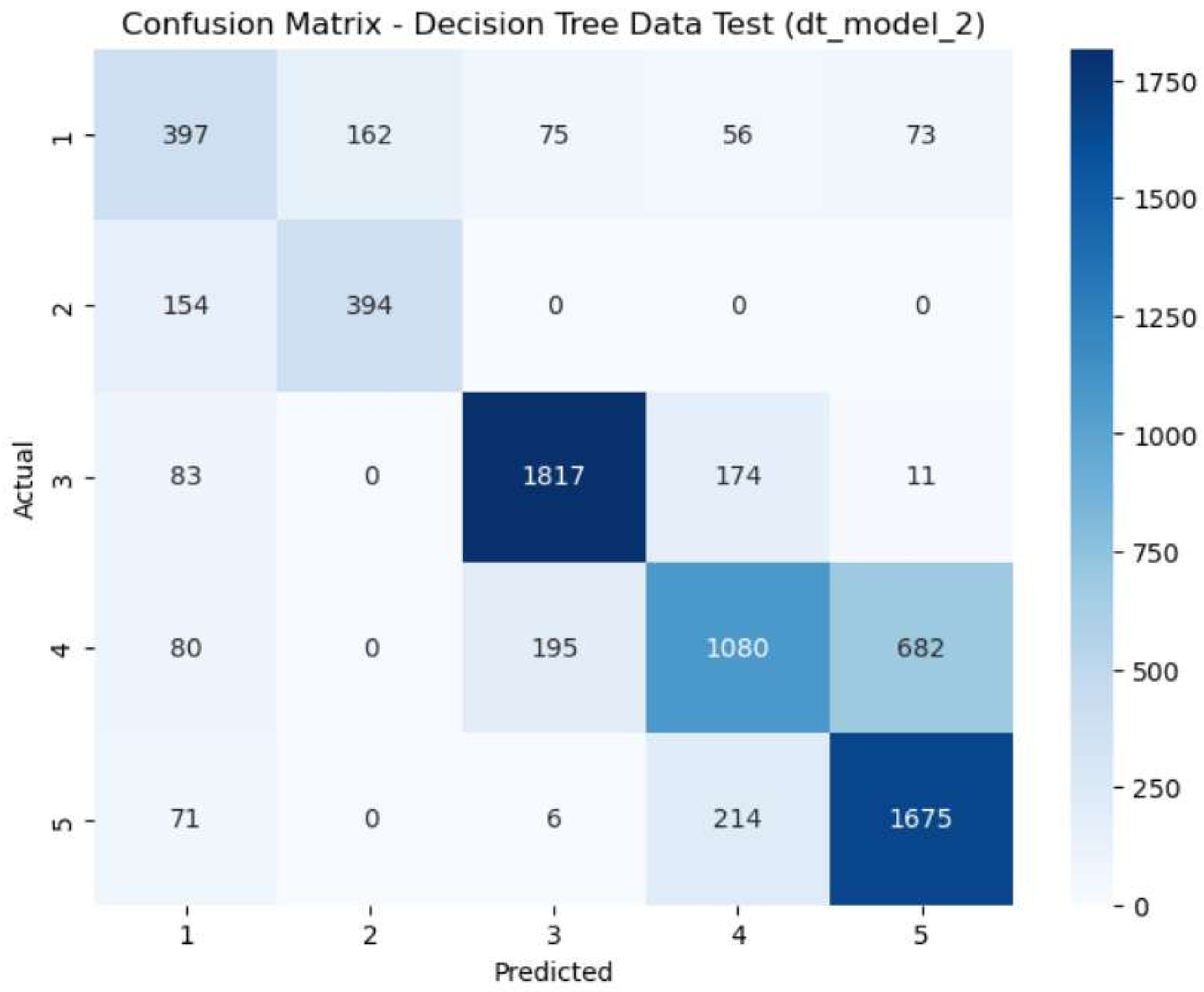


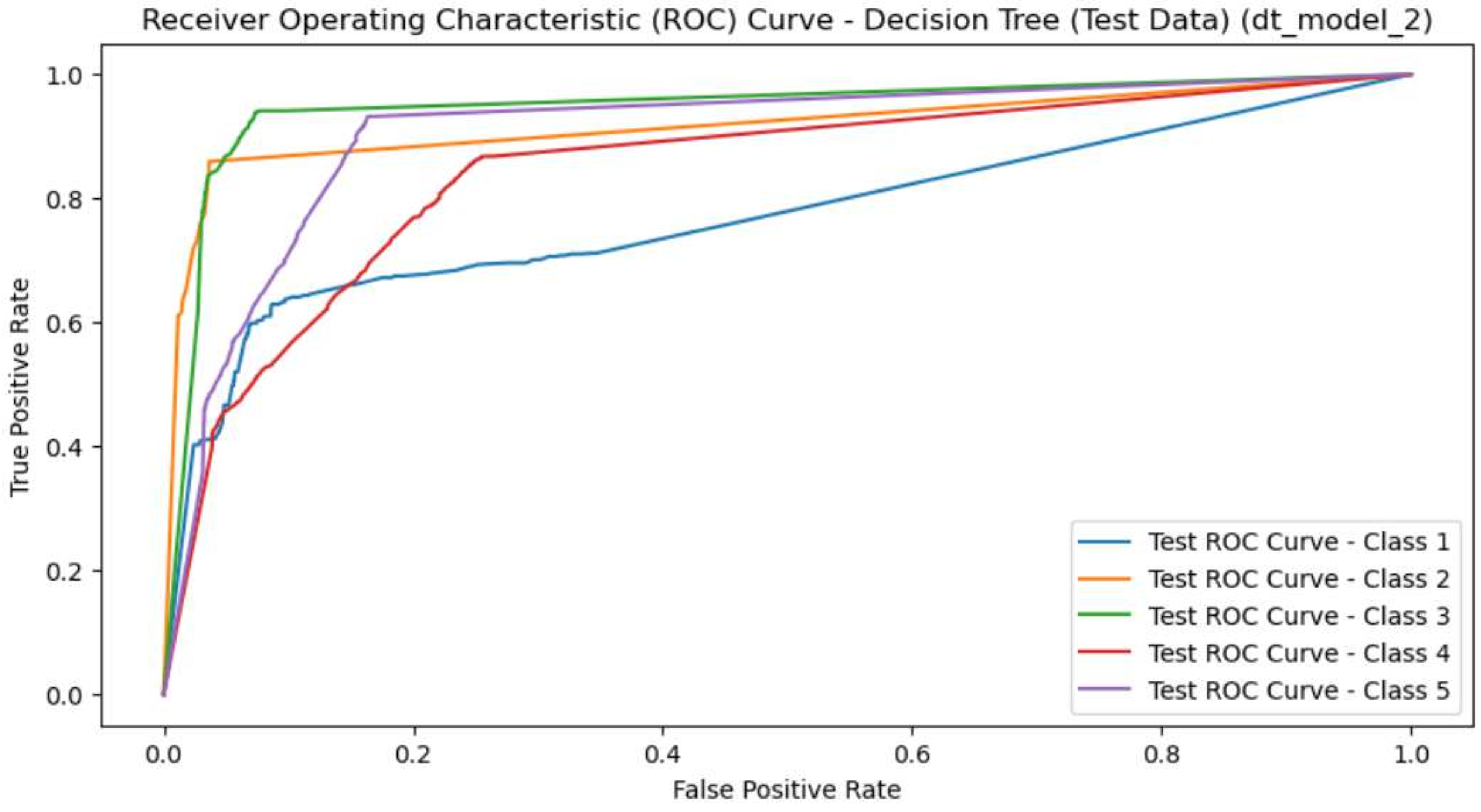


Model with Label Encoder Only

Decision Tree

Model	Accuracy	Precision	Recall	F1 Score	F2 Score
Decision Tree Train (dt_model_2)	0.8634	0.8704	0.8672	0.8640	0.8648
Decision Tree Test (dt_model_2)	0.7248	0.6955	0.6986	0.6926	0.6952

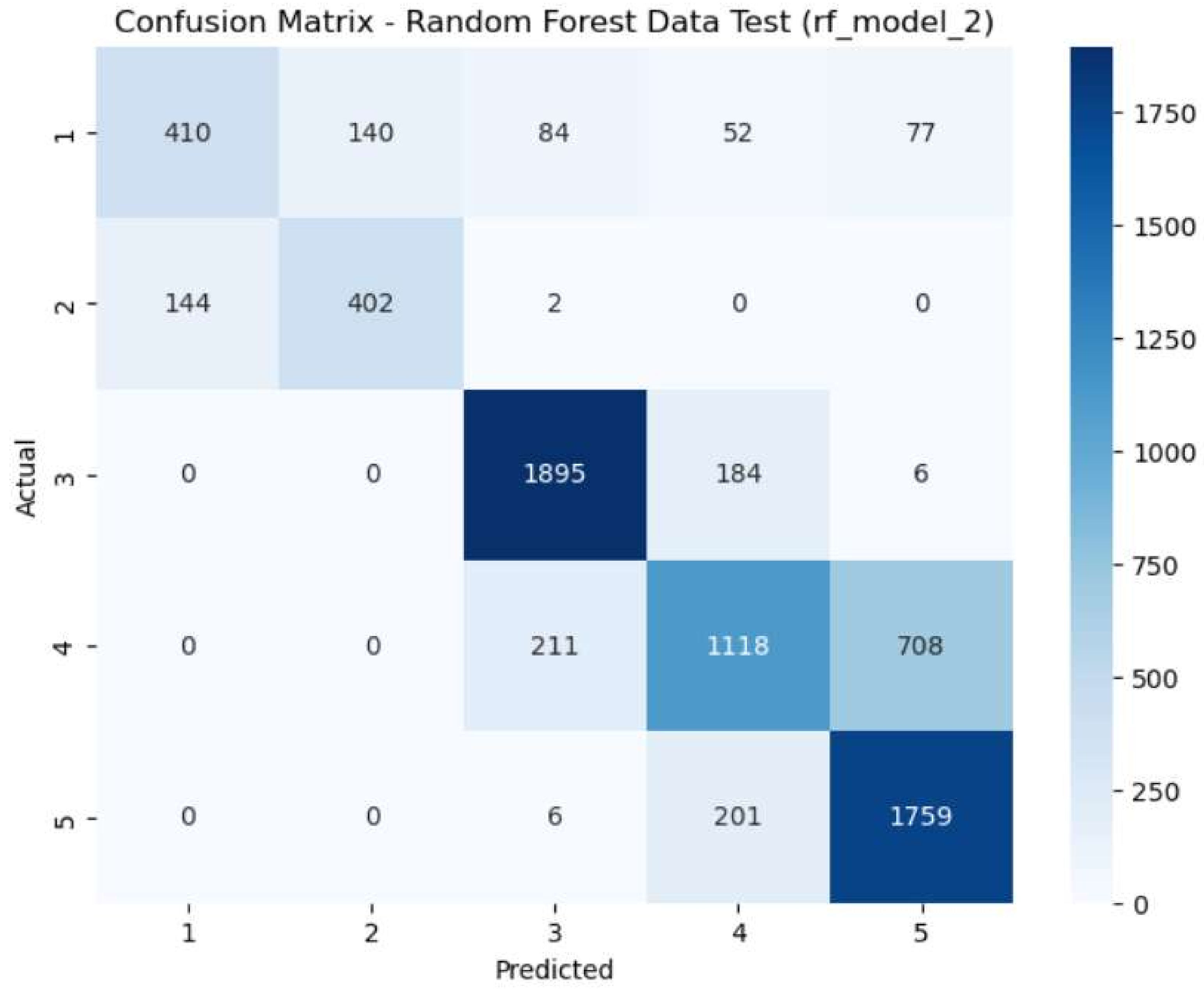


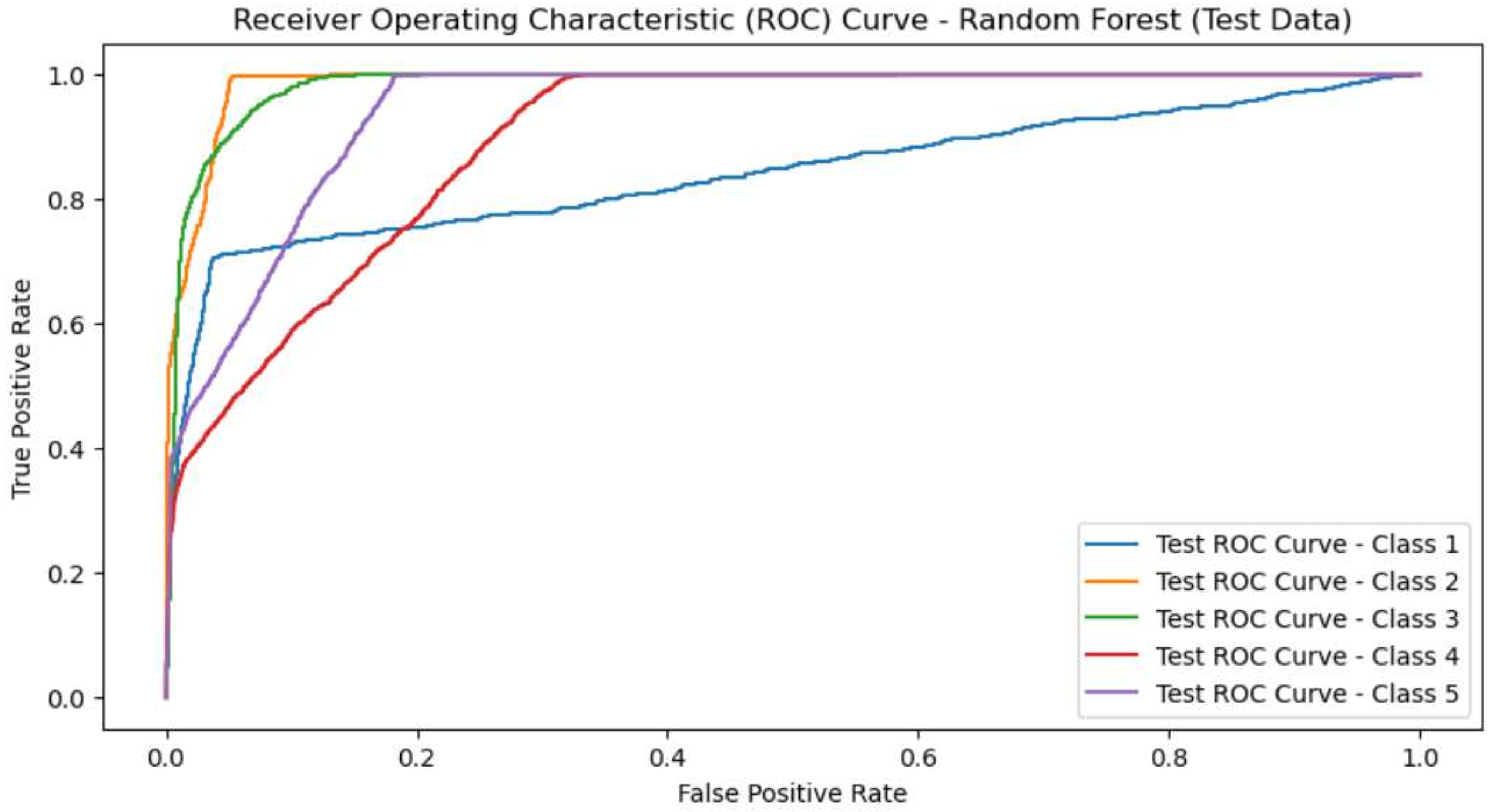


Model with Label Encoder Only

Random Forest

	Accuracy	Precision	Recall	F1 Score	F2 Score
Random Forest Train (rf_model_2)	0.9460	0.9593	0.9367	0.9454	0.9396
Random Forest Test (rf_model_2)	0.7547	0.7505	0.7247	0.7293	0.7247

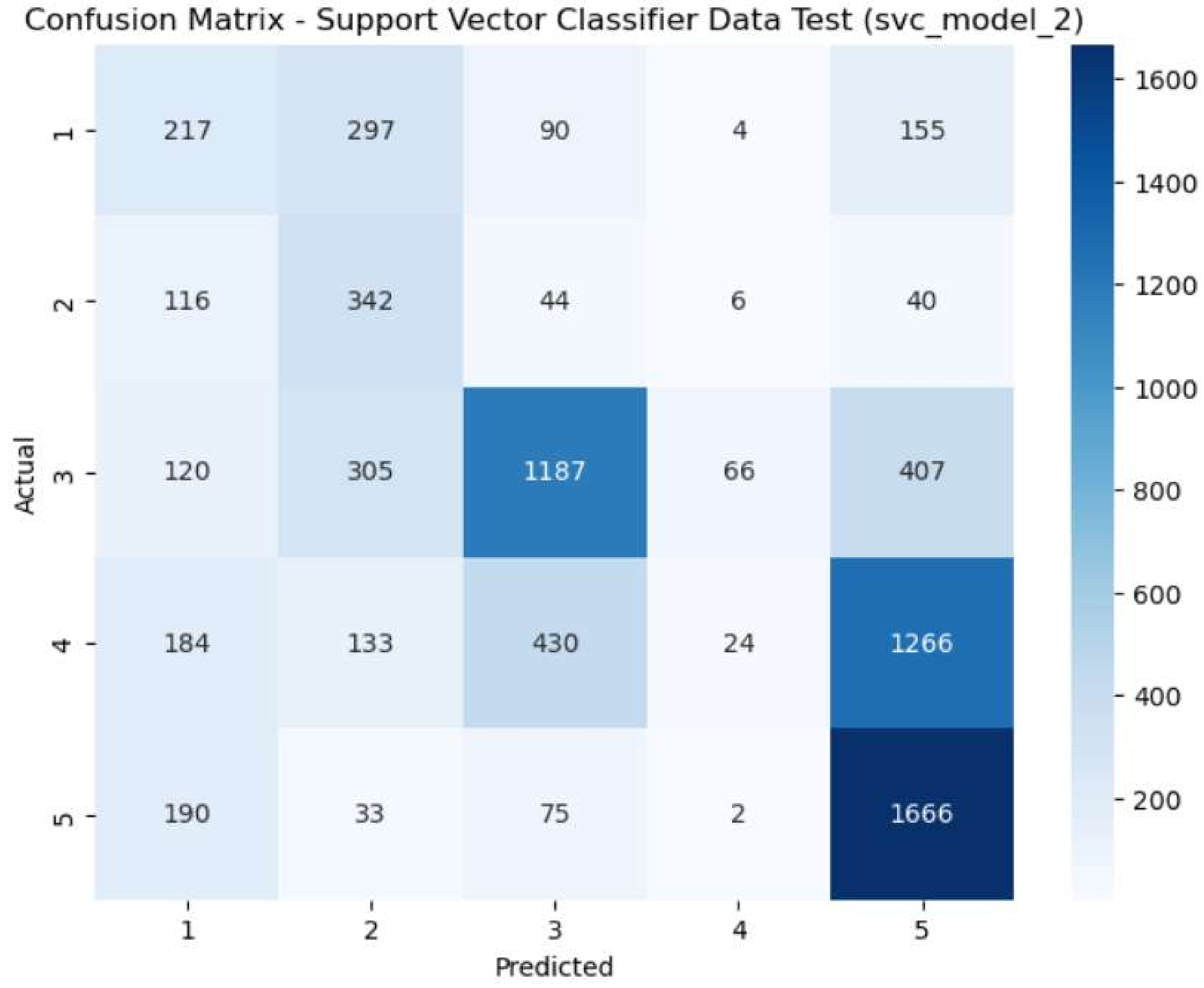


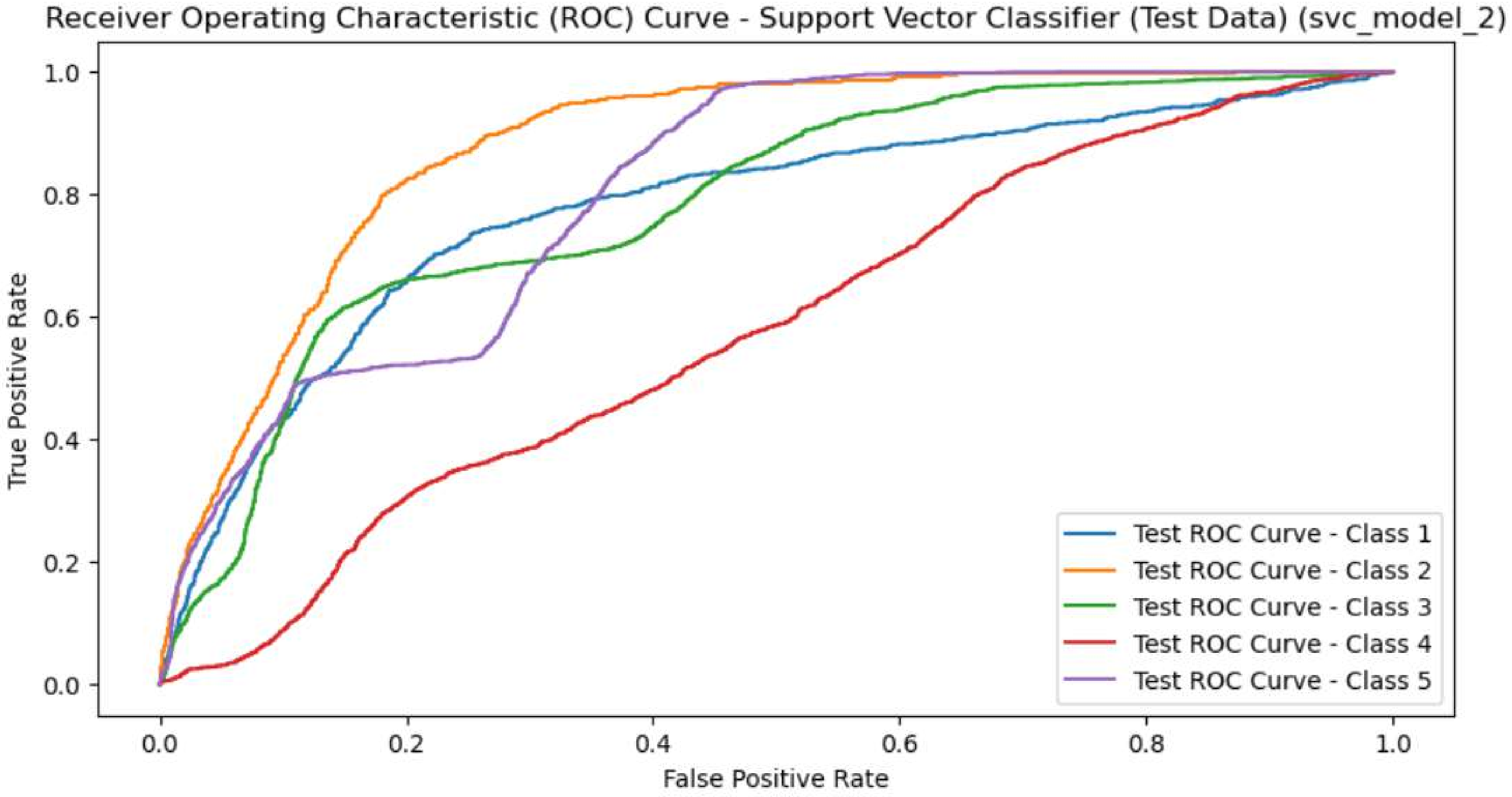


Model with Label Encoder Only

SVM

Model	Accuracy	Precision	Recall	F1 Score	F2 Score
Support Vector Classifier Train (svc_model_2)	0.4632	0.3871	0.4684	0.3832	0.4251
Support Vector Classifier Test (svc_model_2)	0.4644	0.3855	0.4674	0.3842	0.4254

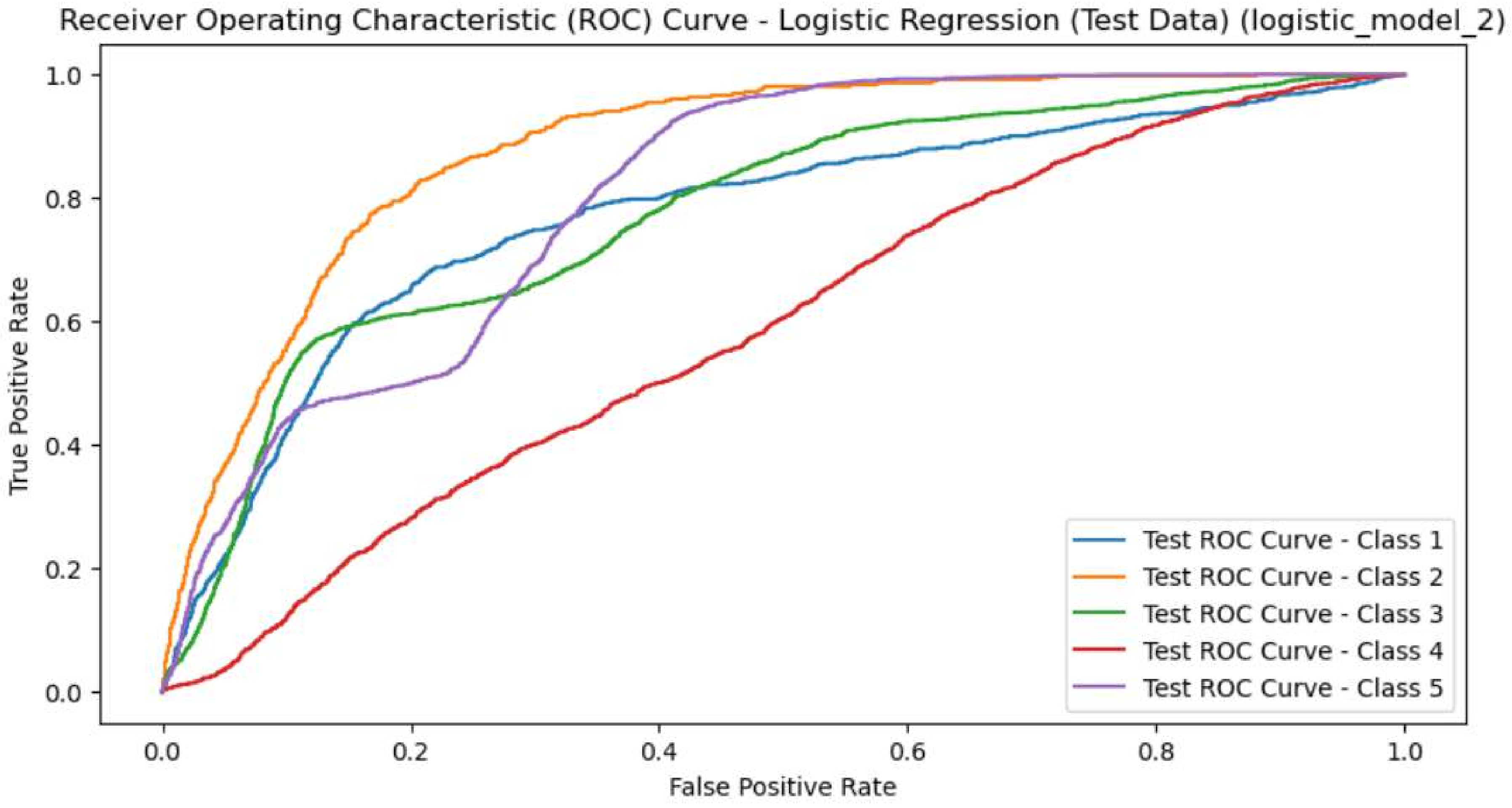




Model with Label Encoder Only

Logistic Regression

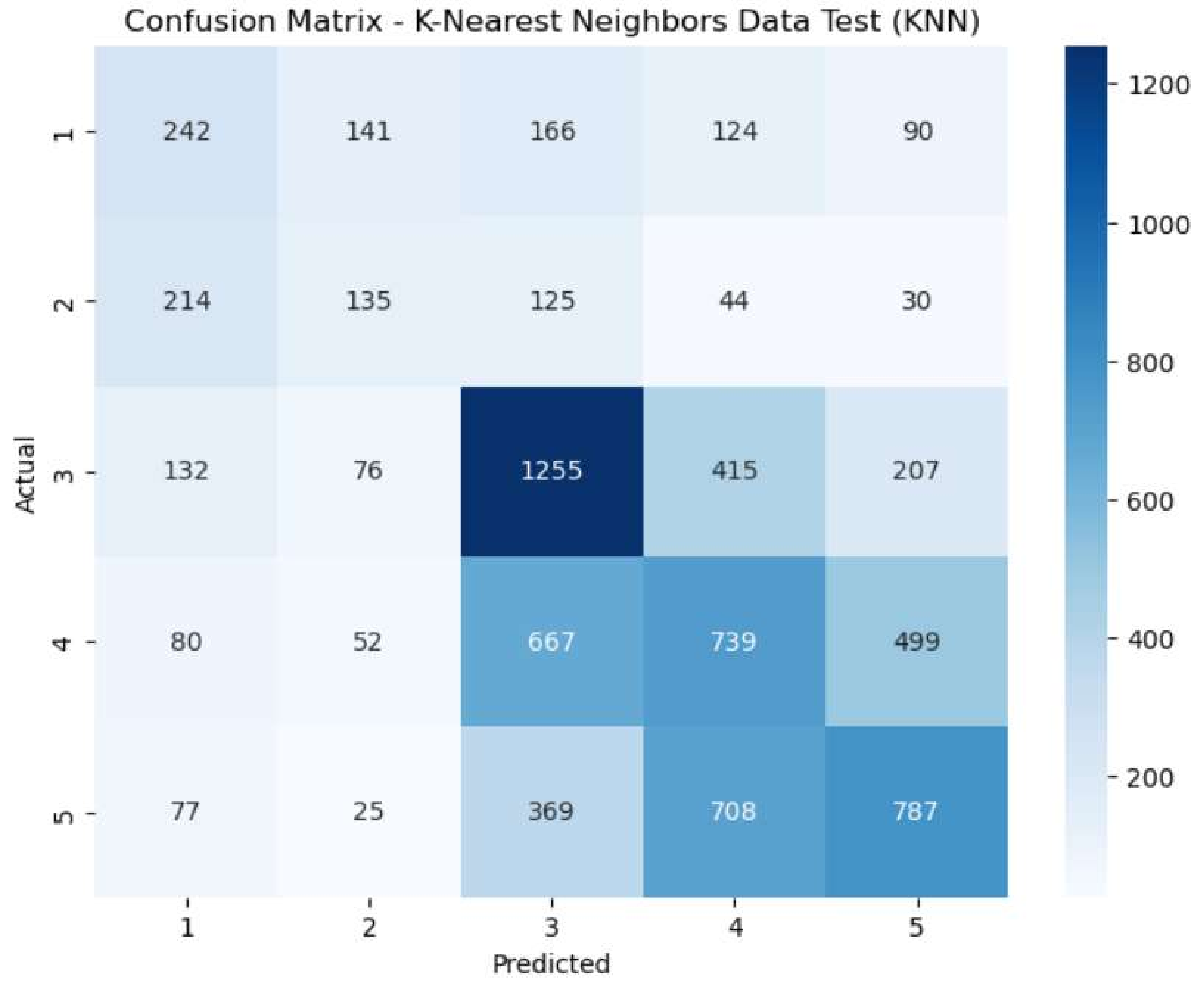
Model	Accuracy	Precision	Recall	F1 Score	F2 Score
Logistic Regression Train (logistic_model_2)	0.4182	0.3865	0.4328	0.3946	0.4127
Logistic Regression Test (logistic_model_2)	0.4229	0.3875	0.4337	0.3959	0.4139

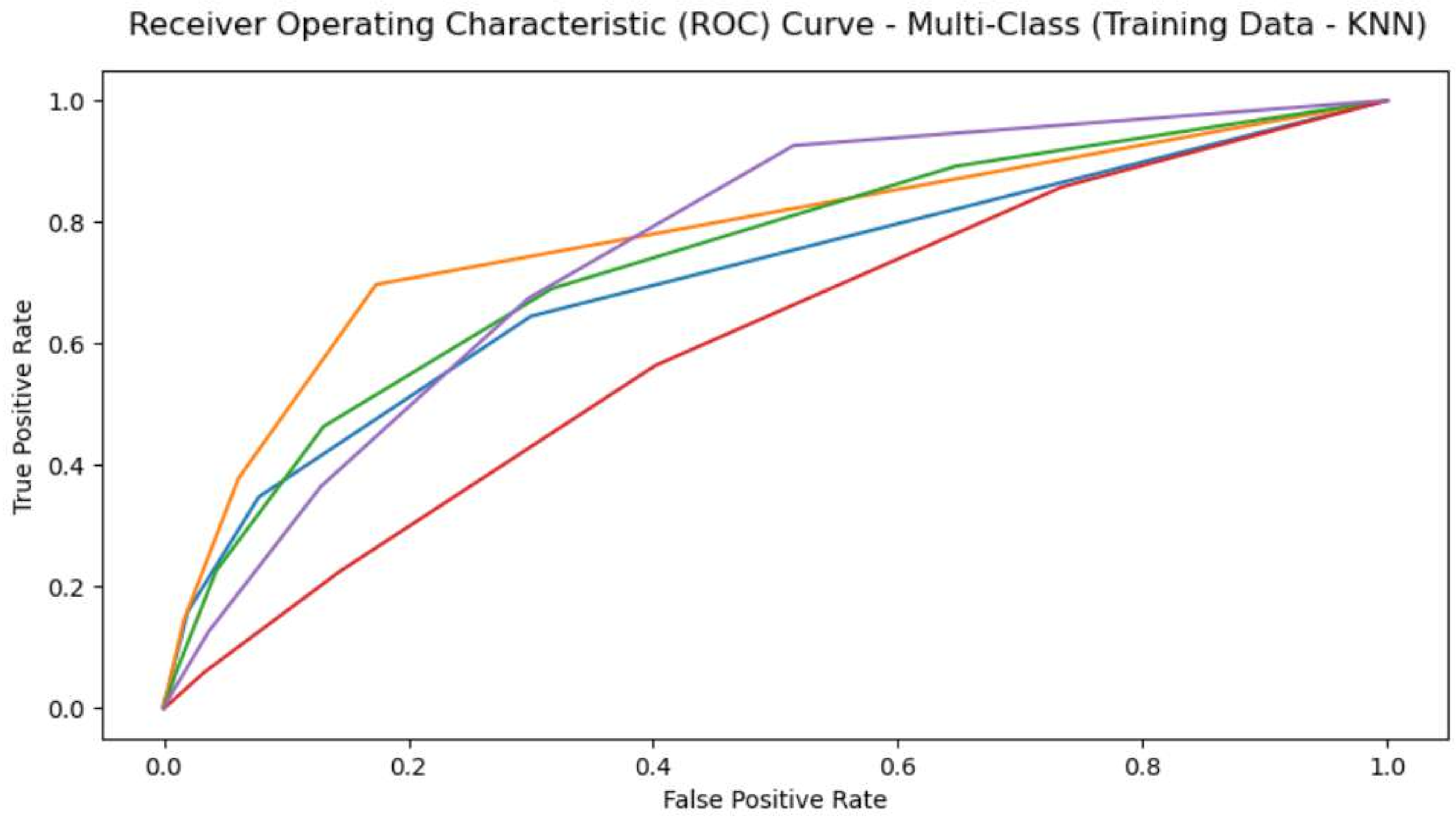


Model with Label Encoder Only

KNN

	Accuracy	Precision	Recall	F1 Score	F2 Score
KNN Train	0.6320	0.6245	0.6028	0.6105	0.6051
KNN Test	0.4268	0.3955	0.3857	0.3877	0.3858





Compare the Model

Decision Tree

One hot

	Accuracy	Precision	Recall	F1 Score	F2 Score
Data Train	0.8347	0.8581	0.8431	0.8414	0.8403
Data Test	0.7443	0.7250	0.7193	0.7152	0.7160

Label

Accuracy	Precision	Recall	F1 Score	F2 Score
0.8634	0.8704	0.8672	0.8640	0.8648
0.7248	0.6955	0.6986	0.6926	0.6952

Random Forest

	Accuracy	Precision	Recall	F1 Score	F2 Score
Data Train	0.8950	0.9255	0.8915	0.9019	0.8942
Data Test	0.7551	0.7515	0.7266	0.7290	0.7254

Accuracy	Precision	Recall	F1 Score	F2 Score
0.9460	0.9593	0.9367	0.9454	0.9396
0.7547	0.7505	0.7247	0.7293	0.7247

SVM

	Accuracy	Precision	Recall	F1 Score	F2 Score
SVM Train (SVC)	0.5437	0.5056	0.5412	0.4670	0.5014
SVM Test (SVC)	0.5464	0.5159	0.5441	0.4737	0.5063

Accuracy	Precision	Recall	F1 Score	F2 Score
0.4632	0.3871	0.4684	0.3832	0.4251
0.4644	0.3855	0.4674	0.3842	0.4254

Logistic Regression

	Accuracy	Precision	Recall	F1 Score	F2 Score
Logistic Regression Train	0.4893	0.5041	0.4983	0.4782	0.4863
Logistic Regression Test	0.4947	0.5095	0.5026	0.4851	0.4920

Accuracy	Precision	Recall	F1 Score	F2 Score
0.4182	0.3865	0.4328	0.3946	0.4127
0.4229	0.3875	0.4337	0.3959	0.4139

KNN

	Accuracy	Precision	Recall	F1 Score	F2 Score
KNN Train	0.6402	0.6571	0.6356	0.6425	0.6374
KNN Test	0.4526	0.4644	0.4528	0.4547	0.4526

Accuracy	Precision	Recall	F1 Score	F2 Score
0.6320	0.6245	0.6028	0.6105	0.6051
0.4268	0.3955	0.3857	0.3877	0.3858

Hyperparameter Tuning Analysis

Hyper parameter -> n_estimators=100, **max_depth=15** , min_samples_split=5, min_samples_leaf=1

	Accuracy	Precision	Recall	F1 Score	F2 Score
Random Forest Train (rf_model_2a)	0.9460	0.9593	0.9367	0.9454	0.9396
Random Forest Test (rf_model_2a)	0.7547	0.7505	0.7247	0.7293	0.7247

Hyper parameter -> n_estimators=100, **max_depth=20** , min_samples_split=5, min_samples_leaf=1

	Accuracy	Precision	Recall	F1 Score	F2 Score
Random Forest Train (rf_model_2b)	0.9941	0.9950	0.9908	0.9928	0.9916
Random Forest Test (rf_model_2b)	0.7529	0.7490	0.7220	0.7309	0.7245

Hyper parameter -> n_estimators=100, **max_depth=25** , min_samples_split=5, min_samples_leaf=1

	Accuracy	Precision	Recall	F1 Score	F2 Score
Random Forest Train (rf_model_2c)	0.9965	0.9966	0.9942	0.9954	0.9947
Random Forest Test (rf_model_2c)	0.7500	0.7464	0.7179	0.7283	0.7212

Hyper parameter -> **n_estimators=200**, max_depth=15, min_samples_split=5, min_samples_leaf=1

	Accuracy	Precision	Recall	F1 Score	F2 Score
Random Forest Train (rf_model_2d)	0.9943	0.9949	0.9905	0.9926	0.9913
Random Forest Test (rf_model_2d)	0.7524	0.7470	0.7206	0.7287	0.7226

Hyper parameter -> **n_estimators=300**, max_depth=15, min_samples_split=5, min_samples_leaf=1

	Accuracy	Precision	Recall	F1 Score	F2 Score
Random Forest Train (rf_model_2e)	0.9491	0.9624	0.9389	0.9481	0.9420
Random Forest Test (rf_model_2e)	0.7532	0.7495	0.7234	0.7279	0.7233

The Best Model

- Berdasarkan evaluasi model menggunakan metric accuracy, precision, recall, F1&F2 score, confusion matrix dan grafik ROC, model machine learning terbaik yaitu Random Forest dengan menggunakan Label Encoding.

- Setelah mendapatkan model terbaik dilakukan hyperparameter tuning untuk optimisasi model, dan telah diperoleh bahwa model dengan jumlah n estimator 100 dan maksimal depth tree nya adalah 15. Karena jika di tinjau lebih detail model dengan depth tree 20 saja sudah dapat dikatakan overfitting.

Model Deployment

Model Deployment

Deploy the Model with Streamlit

```
import streamlit as st
import numpy as np
import pickle

# Nilai-nilai yang mungkin ada dalam setiap kolom
gender_options = ['F', 'M']
region_category_options = ['Village', 'City', 'Town']
membership_category_options = ['Platinum Membership', 'Premium Membership', 'No Membership', 'Gol
joined_through_referral_options = ['No', 'Yes']
used_special_discount_options = ['Yes', 'No']
offer_application_preference_options = ['Yes', 'No']
past_complaint_options = ['No', 'Yes']
complaint_status_options = ['Not Applicable', 'Solved', 'Solved in Follow-up', 'Unsolved', 'No In
feedback_options = ['Products always in Stock', 'Quality Customer Care', 'Poor Website', 'No reas
```

```
# Load your pre-trained models and scalers
with open('model/scaler.pkl', 'rb') as scaler_file:
    scaler = pickle.load(scaler_file)

with open('model/log_transformed_ATV.pkl', 'rb') as log_file:
    log_transformed_ATV = pickle.load(log_file)

with open('model/log_transformed_PIW.pkl', 'rb') as log_file:
    log_transformed_PIW = pickle.load(log_file)

with open('model/log_transformed_ATS.pkl', 'rb') as log_file:
    log_transformed_ATS = pickle.load(log_file)

with open('model/random_forest_model.pkl', 'rb') as model_file:
    rf_model_2a = pickle.load(model_file)
```

Model Deployment

This is the User Inteface

```python
# Streamlit UI
st.title("Churn Risk Score Prediction for HackelEarth Website")

# Create input fields for user data
customer_id = st.text_input('Customer ID', '')
age = st.slider('Age', 10, 100, 25)
gender = st.selectbox('Gender', gender_options)
region_category = st.selectbox('Region Category', region_category_options)
membership_category = st.selectbox('Membership Category', membership_category_options)
joined_through_referral = st.selectbox('Joined Through Referral', joined_through_referral_options)
avg_time_spent = st.number_input('Average Time Spent', min_value=0)
avg_transaction_value = st.number_input('Average Transaction Value', min_value=0)
avg_frequency_login_days = st.number_input('Average Frequency Login Days', min_value=0)
points_in_wallet = st.number_input('Points in Wallet', min_value=0)
used_special_discount = st.selectbox('Used Special Discount', used_special_discount_options)
offer_application_preference = st.selectbox('Offer Application Preference', offer_application_preference_options)
past_complaint = st.selectbox('Past Complaint', past_complaint_options)
complaint_status = st.selectbox('Complaint Status', complaint_status_options)
feedback = st.selectbox('Feedback', feedback_options)
```

Model Deployment

Converting categorical data to numerical data

```
# Make prediction when the user clicks the "Submit" button
if st.button('Submit'):
    # Prepare input data

    # Apply LabelEncoder to membership_category
    MC =  0 if membership_category== 'Basic Membership' else (1 if membership_category=='Gold Membership'
                                                           else (2 if membership_category=='No Membership'
                                                                 else(3 if membership_category=='Platinum Membership'
                                                                      else(4 if membership_category=='Premium Membership'
                                                                           else 5 ))))

    # Apply LabelEncoder to gender
    GEN = 0 if gender== 'F' else 1

    # Apply LabelEncoder to region_category
    RC = 0 if region_category=='City'else (1 if region_category=='Town'else 2)

    # Apply LabelEncoder to joined_through_referral
    JTR = 0 if joined_through_referral=='No' else 1

    # Apply LabelEncoder to used_special_discount
    USD = 0 if used_special_discount=='No' else 1

    # Apply LabelEncoder to offer_application_preference
    OAP = 0 if offer_application_preference=='No' else 1

        # Apply LabelEncoder to past_complaint
    PC = 0 if past_complaint=='No' else 1
```

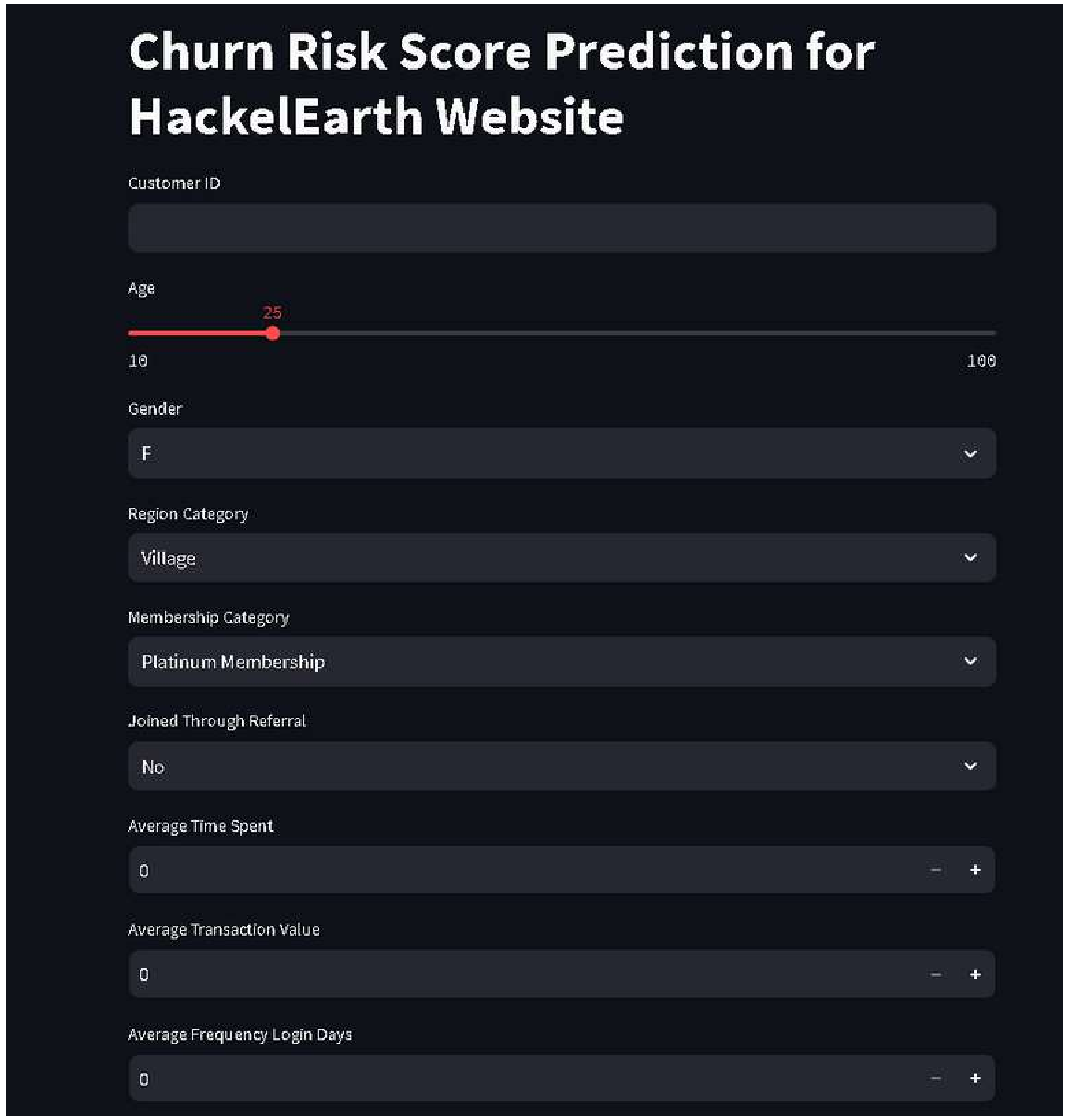
Model Deployment

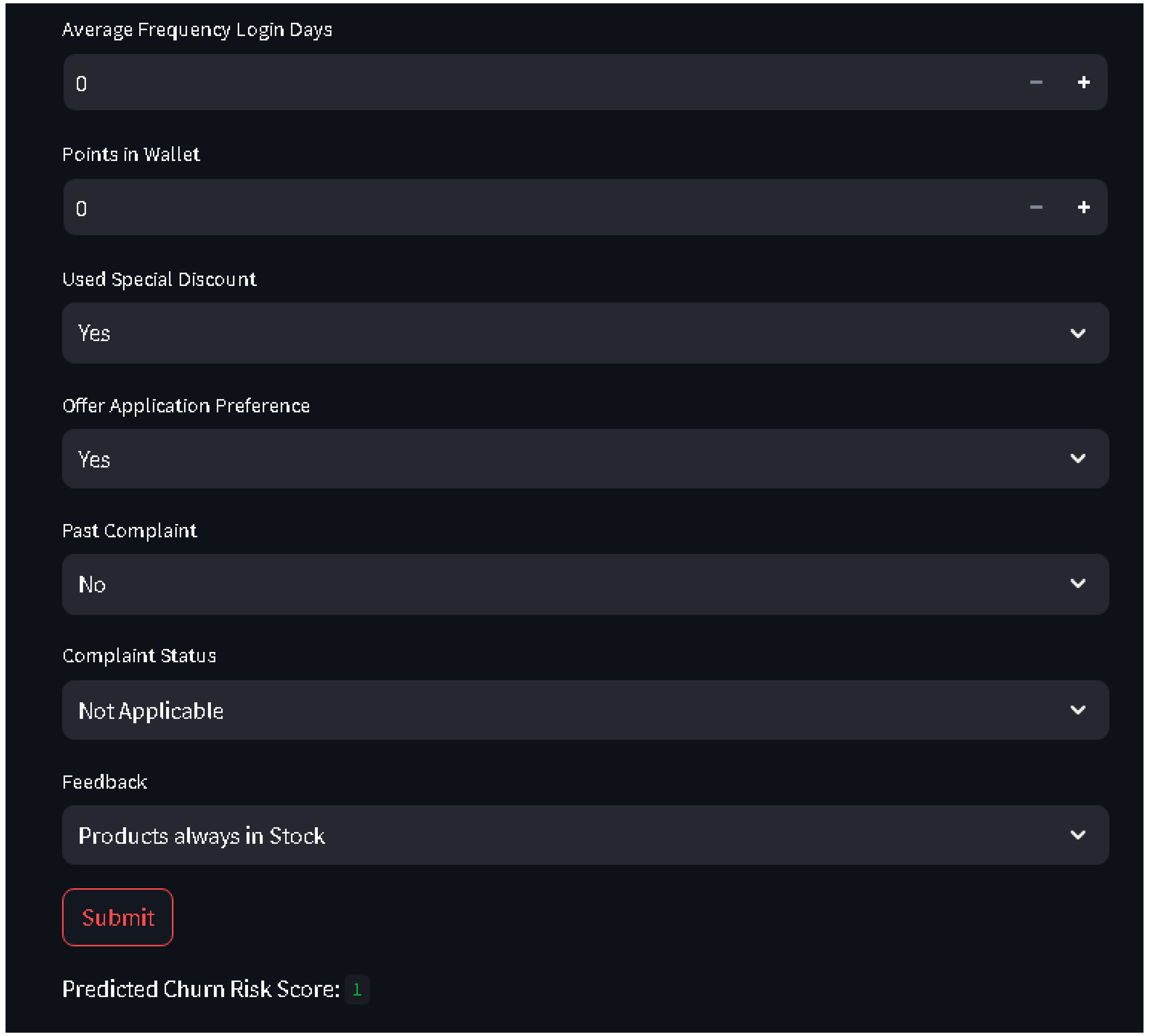
Scaling the data, and Perform Prediction with Random Forest Model

```
# Combine input data
input_data=[age, GEN, RC, MC,JTR, avg_frequency_login_days, USD, OAP, PC, CS,
            fb,points_in_wallet, avg_time_spent, avg_transaction_value]

scaled_input_data = scaler.transform(np.array([input_data]))



# Perform prediction with the Random Forest model
predicted_churn_risk_score = rf_model_2a.predict(scaled_input_data)[0]

# Display the predicted Churn Risk Score to the user
st.write('Predicted Churn Risk Score:', predicted_churn_risk_score)
```

Result

Streamlit Link ⟶ https://fc45f26r9zptzfuzxlhta8.streamlit.app/





Conclusion

1. Berdasarkan visualisai distribusi variabel kategoris dan heatmap correlation, dapat diambil wawasan bahwa fitur yang memiliki peran besar dalam hasil prediksi churn rate score adalah membership category, feedback, avg transaction value, dan points in wallet.
2. Pemodelan berjalan lebih efektif pada dataset yang dilakukan label encoder saja
3. Dari proses pemodelan yang dilakukan pada dataset churn, pemodelan yang paling cocok digunakan untuk memprediksi churn rate score adalah pemodelan dengan algoritma Random Forest
4. Pada pemodelan Random Forest didapatkan nilai akurasi: 94%, Precision : 95%, dan Recall: 93% untuk Data Train
5. Pada pemodelan Random Forest didapatkan nilai akurasi: 75%, Precision : 75%, dan Recall: 72% untuk Data Test
6. Dengan n_estimator = 100, max_depth = 15, min_samples_spill= 5, min samples_leaf = 1 didapatkan nilai akurasi: 94%, Precision : 95%, dan Recall: 93% untuk Data Train
7. Dengan n_estimator = 100, max_depth = 15, min_samples_spill= 5, min samples_leaf = 1 didapatkan nilai akurasi: 75%, Precision : 75%, dan Recall: 72% untuk Data Test
8. Model akan lebih bagus jika dilakukan cross validation dan Grid Search untuk hyperparameter tuning.

Thank you

Any Questions?

Pembagian Tugas

Fajar: EDA & Data Pre-Processing

Melani: Modelling & Model Deployment