loan-eligibility-prediction

October 9, 2024

1 Loan Eligibility Prediction

2 Importing Libraries

[1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from imblearn.over_sampling import SMOTE  # For oversampling
import matplotlib.pyplot as plt
import seaborn as sns
```

3 Loading datasets

[2]:
```
credit_card_info = pd.read_csv('C:/Users/ASUS/Desktop/Power BI Practice/credit␣
↪card fruad/cc_info.csv')
transaction_data = pd.read_csv('C:/Users/ASUS/Desktop/Power BI Practice/credit␣
↪card fruad/transactions.csv')
```

[3]:
```
credit_card_info.head()
```

[3]:

	credit_card	city	state	zipcode	credit_card_limit
0	1280981422329509	Dallas	PA	18612	6000
1	9737219864179988	Houston	PA	15342	16000
2	4749889059323202	Auburn	MA	1501	14000
3	9591503562024072	Orlando	WV	26412	18000
4	2095640259001271	New York	NY	10001	20000

[4]:
```
transaction_data.head()
```

[4]:

	credit_card	date	transaction_dollar_amount	\
0	1003715054175576	2015-09-11 00:32:40	43.78	
1	1003715054175576	2015-10-24 22:23:08	103.15	
2	1003715054175576	2015-10-26 18:19:36	48.55	
3	1003715054175576	2015-10-22 19:41:10	136.18	

```
 4  1003715054175576  2015-10-26 20:08:22                    71.82

        Long        Lat
0 -80.174132  40.267370
1 -80.194240  40.180114
2 -80.211033  40.313004
3 -80.174138  40.290895
4 -80.238720  40.166719
```

[5]:
```
credit_card_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 984 entries, 0 to 983
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   credit_card        984 non-null    int64
 1   city               984 non-null    object
 2   state              984 non-null    object
 3   zipcode            984 non-null    int64
 4   credit_card_limit  984 non-null    int64
dtypes: int64(3), object(2)
memory usage: 38.6+ KB
```

[6]:
```
transaction_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294588 entries, 0 to 294587
Data columns (total 5 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   credit_card               294588 non-null  int64
 1   date                      294588 non-null  object
 2   transaction_dollar_amount  294588 non-null  float64
 3   Long                      294588 non-null  float64
 4   Lat                       294588 non-null  float64
dtypes: float64(3), int64(1), object(1)
memory usage: 11.2+ MB
```

4 Calculating total transactions per credit card

[7]:
```
total_transactions = transaction_data.
  ↪groupby('credit_card')['transaction_dollar_amount'].sum().reset_index()
total_transactions.head()
```

[7]:
```
         credit_card  transaction_dollar_amount
0  1003715054175576                   28839.84
1  1013870087888817                   36814.88
2  1023820165155391                   61052.56
3  1073931538936472                    7406.27
4  1077622576192810                    1790.73
```

5 Merge datasets

[8]:
```
merged_data = pd.merge(credit_card_info, total_transactions, on='credit_card',␣
↪how='left')
merged_data.rename(columns={'transaction_dollar_amount':␣
↪'total_transaction_amount'}, inplace=True)
merged_data['total_transaction_amount'] =␣
↪merged_data['total_transaction_amount'].fillna(0)
```

6 details of Merge data

[9]:
```
merged_data.head()
```

[9]:
```
         credit_card      city state  zipcode  credit_card_limit  \
0  1280981422329509    Dallas    PA    18612               6000
1  9737219864179988   Houston    PA    15342              16000
2  4749889059323202    Auburn    MA     1501              14000
3  9591503562024072   Orlando    WV    26412              18000
4  2095640259001271  New York    NY    10001              20000

   total_transaction_amount
0                  16767.89
1                  44370.56
2                  25128.09
3                  43217.20
4                  48546.94
```

[10]:
```
merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 984 entries, 0 to 983
Data columns (total 6 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   credit_card              984 non-null    int64
 1   city                     984 non-null    object
 2   state                    984 non-null    object
 3   zipcode                  984 non-null    int64
```

```
 4   credit_card_limit         984 non-null    int64
 5   total_transaction_amount  984 non-null    float64
dtypes: float64(1), int64(3), object(2)
memory usage: 46.3+ KB
```

[11]:
```
merged_data.isnull().sum()
```

[11]:
```
credit_card                 0
city                        0
state                       0
zipcode                     0
credit_card_limit           0
total_transaction_amount    0
dtype: int64
```

7 Feature Engineering: Add log-transformed variables

[12]:
```
merged_data['LoanAmount_log'] = np.
↪log1p(merged_data['total_transaction_amount'])  # log1p to handle zero values
merged_data['TotalIncome_log'] = np.log1p(merged_data['credit_card_limit'])  #␣
↪log1p to handle zero values
```

8 Creating target variable 'is_approved'

[13]:
```
merged_data['is_approved'] = (merged_data['total_transaction_amount'] <=␣
↪merged_data['credit_card_limit']).astype(int)
```

9 Define features (X) and target (y)

[14]:
```
X = merged_data[['credit_card_limit', 'total_transaction_amount',␣
↪'LoanAmount_log', 'TotalIncome_log']]
y = merged_data['is_approved']
```

10 Spliting the data into training and testing sets

[15]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42, stratify=y)
```

11 Handling class imbalance using SMOTE (Synthetic Minority Oversampling Technique)

[16]:
```
smote = SMOTE(random_state=42)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)
```

12 Train a Random Forest Classifier

[17]:
```
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train_res, y_train_res)
```

[17]:
```
RandomForestClassifier(random_state=42)
```

13 Making predictions on the test set

[18]:
```
y_pred = rf_model.predict(X_test)
```

14 Evaluating the model

[19]:
```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.99
```

15 Classification report

[20]:
```
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=['Not Approved',␣
↪'Approved']))
```

```
Classification Report:
              precision    recall  f1-score   support

Not Approved       1.00      1.00      1.00       255
    Approved       0.98      0.98      0.98        41

    accuracy                           0.99       296
   macro avg       0.99      0.99      0.99       296
weighted avg       0.99      0.99      0.99       296
```

16 Plotting the approval status in a pie chart

[21]:
```
approved_count = sum(y_pred)
unapproved_count = len(y_pred) - approved_count
```

17 Print the counts

[22]:
```
print("Number of Approved Loans (1):", sum(y_pred))
print("Number of Unapproved Loans (0):", len(y_pred)-sum(y_pred))
```

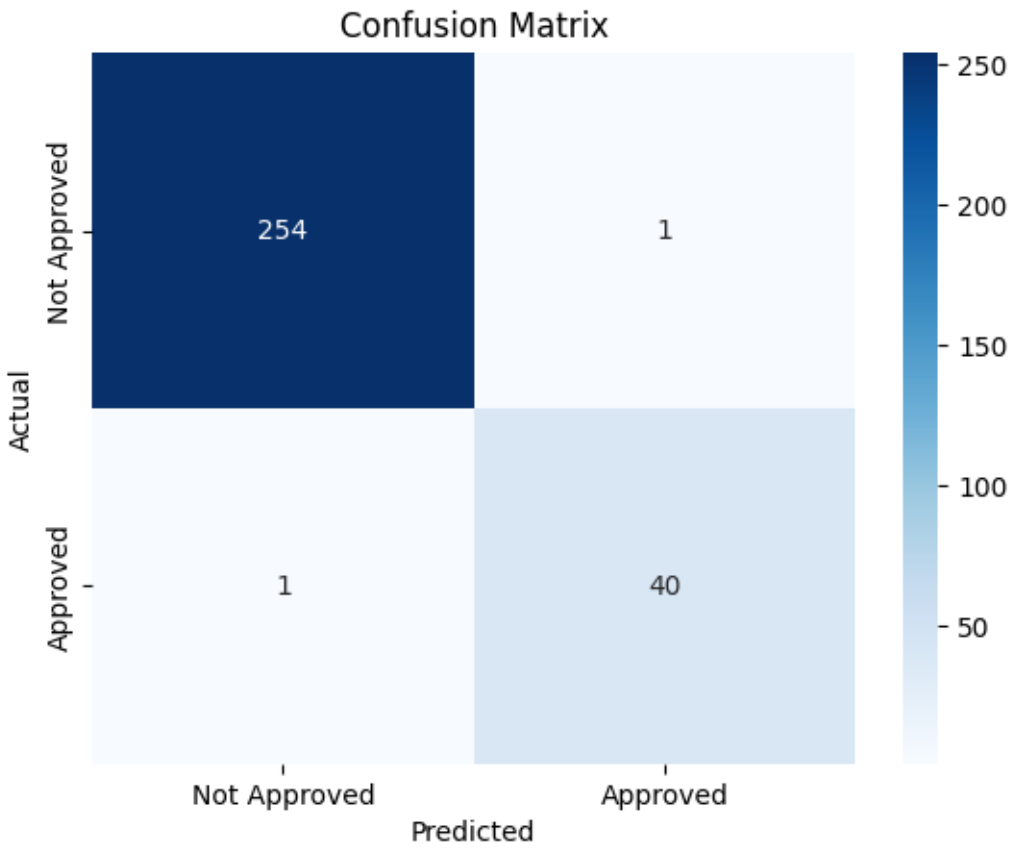
```
Number of Approved Loans (1): 41
Number of Unapproved Loans (0): 255
```

18 Plot Confusion Matrix

[23]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=['Not Approved',␣
 ↪'Approved'], yticklabels=['Not Approved', 'Approved'])
print(cm)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

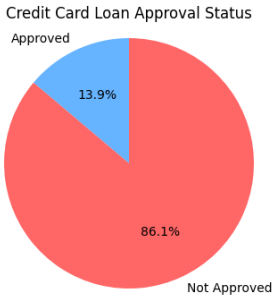
```
[[254   1]
 [  1  40]]
```



19 Plotting the approval status in a pie chart

[24]:
```
approved_count = sum(y_pred)
unapproved_count = len(y_pred) - approved_count
import matplotlib.pyplot as plt
labels = ['Approved', 'Not Approved']
sizes = [approved_count, unapproved_count]
colors = ['#66b3ff', '#ff6666']

plt.figure(figsize=(16, 4))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90)
plt.title('Credit Card Loan Approval Status')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```


Credit Card Loan Approval Status
Approved
13.9%
86.1%
Not Approved