Untitled2

July 26, 2025

[4]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
```

[7]:
```
import pandas as pd
customers = pd.read_csv("train_customers.csv")
restaurants = pd.read_csv("vendors.csv")
orders = pd.read_csv("orders.csv")
test_data = pd.read_csv("train_locations.csv")
```

```
/tmp/ipykernel_297/337063056.py:4: DtypeWarning: Columns (15,16,18,19,20) have
mixed types. Specify dtype option on import or set low_memory=False.
  orders = pd.read_csv("orders.csv")
```

[8]:
```
print("Customers:\n", customers.head())
print("Vendors:\n", restaurants.head())
print("Orders:\n", orders.head())
```

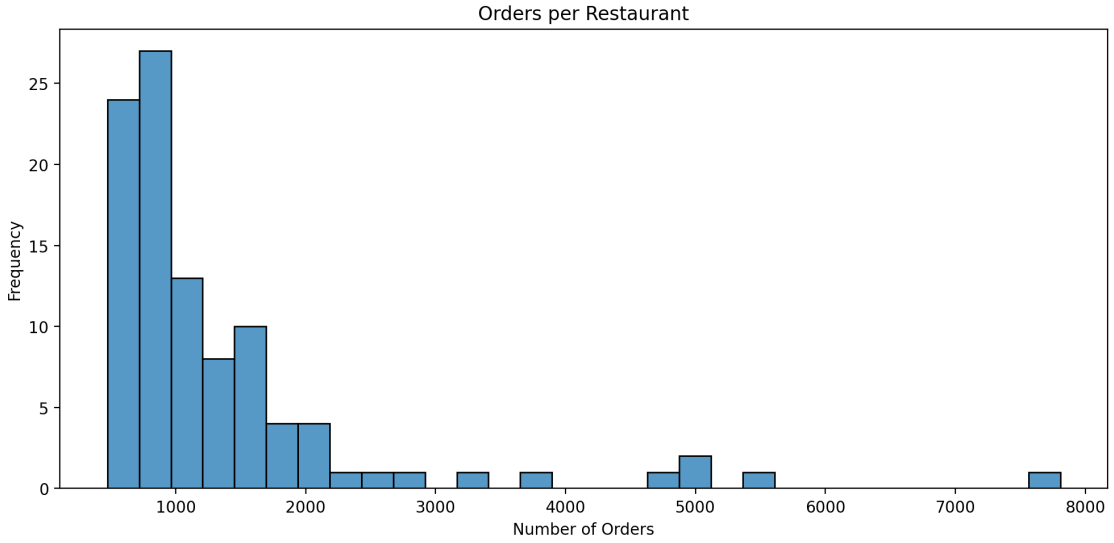
```
Customers:
   customer_id gender  dob  status  verified language       created_at  \
0     TCHWPBT   Male  NaN       1         1       EN   2/7/2023 19:16
1     ZGFSYCZ   Male  NaN       1         1       EN   2/9/2023 12:04
2     S2ALZFL   Male  NaN       0         1       EN  3/14/2023 18:31
3     952DBJQ   Male  NaN       1         1       EN  3/15/2023 19:47
4     1IX6FXS   Male  NaN       1         1       EN  3/15/2023 19:57

        updated_at
0   2/7/2023 19:16
1   2/9/2023 12:04
2  3/14/2023 18:31
3  3/15/2023 19:47
4  3/15/2023 19:57
Vendors:
```

```
    id  authentication_id  latitude  longitude vendor_category_en  \
0   4             118597 -0.588596   0.754434        Restaurants
1  13             118608 -0.471654   0.744470        Restaurants
2  20             118616 -0.407527   0.643681        Restaurants
3  23             118619 -0.585385   0.753811        Restaurants
4  28             118624  0.480602   0.552850        Restaurants

   vendor_category_id  delivery_charge  serving_distance  is_open  \
0                   2              0.0                 6        1
1                   2              0.7                 5        1
2                   2              0.0                 8        1
3                   2              0.0                 5        1
4                   2              0.7                15        1

       OpeningTime  … open_close_flags               vendor_tag  \
0  11:00AM-11:30PM  …                1  2,4,5,8,91,22,12,24,16,23
1  08:30AM-10:30PM  …                1  4,41,51,34,27,15,24,16,28
2  08:00AM-10:45PM  …                1                4,8,91,10
3  10:59AM-10:30PM  …                1                5,8,30,24
4  11:00AM-11:45PM  …                1                        5

                                  vendor_tag_name one_click_vendor  \
0  Arabic,Breakfast,Burgers,Desserts,Free Deliver…                Y
1  Breakfast,Cakes,Crepes,Italian,Pasta,Pizzas,Sa…                Y
2           Breakfast,Desserts,Free Delivery,Indian                Y
3                    Burgers,Desserts,Fries,Salads                Y
4                                           Burgers                Y

   country_id  city_id       created_at      updated_at device_type  \
0           1        1  1/30/2023 14:42  4/7/2025 15:12           3
1           1        1   5/3/2023 12:32  4/5/2025 20:46           3
2           1        1   5/4/2023 22:28  4/7/2025 16:35           3
3           1        1   5/6/2023 19:20   4/2/2025 0:56           3
4           1        1  5/17/2023 22:12  4/5/2025 15:57           3

   display_orders
0               1
1               1
2               1
3               1
4               1

[5 rows x 59 columns]
Orders:
    order_id customer_id  item_count  grand_total  payment_mode promo_code  \
0  163923.0     KL09J9N         6.0         10.1             1        NaN
1  163924.0     H5LGGFX         3.0          8.4             1        NaN
2  163925.0     CYLZB6T         4.0         15.0             1        NaN
```

```
3  163929.0     4YKUKYN         7.0          27.2             1        NaN
4  163930.0     WDNU30K         1.0           6.5             1        NaN

   vendor_discount_amount  promo_code_discount_percentage is_favorite  \
0                     0.0                             NaN         NaN
1                     0.0                             NaN         NaN
2                     0.0                             NaN         NaN
3                     0.0                             NaN         NaN
4                     0.0                             NaN         NaN

  is_rated  …  driver_accepted_time  ready_for_pickup_time  picked_up_time  \
0       No  …                   NaN                    NaN             NaN
1       No  …                   NaN                    NaN             NaN
2       No  …                   NaN                    NaN             NaN
3       No  …                   NaN                    NaN             NaN
4       No  …                   NaN                    NaN             NaN

   delivered_time  delivery_date vendor_id     created_at LOCATION_NUMBER  \
0             NaN  8/1/2024 5:30        84  8/2/2024 5:33               0
1             NaN  8/1/2024 5:30        78  8/2/2024 5:34               0
2             NaN  8/1/2024 5:30         4  8/2/2024 5:35               0
3             NaN  8/1/2024 5:30       157  8/2/2024 5:39               0
4             NaN  8/1/2024 5:30       160  8/2/2024 5:39               0

  LOCATION_TYPE CID X LOC_NUM X VENDOR
0          Work       KL09J9N X 0 X 84
1          Home       H5LGGFX X 0 X 78
2          Work        CYLZB6T X 0 X 4
3          Home      4YKUKYN X 0 X 157
4          Home      WDNU30K X 0 X 160

[5 rows x 26 columns]
```

[11]:
```python
# Count the number of orders for each vendor (restaurant)
order_counts = orders['vendor_id'].value_counts()
import matplotlib.pyplot as plt
import seaborn as sns
# Plot the distribution
plt.figure(figsize=(10, 5))
sns.histplot(order_counts, bins=30, kde=False)
plt.title("Orders per Restaurant")
plt.xlabel("Number of Orders")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```

[11]:



[13]:
```python
import pandas as pd

# Load datasets
orders = pd.read_csv("orders.csv")
customers = pd.read_csv("train_customers.csv")
restaurants = pd.read_csv("vendors.csv")

# Step 0: Clean column names (remove extra spaces)
orders.columns = orders.columns.str.strip()
customers.columns = customers.columns.str.strip()
restaurants.columns = restaurants.columns.str.strip()

# Step 1: Inspect columns
print("Orders columns:", orders.columns.tolist())
print("Customers columns:", customers.columns.tolist())
print("Restaurants columns:", restaurants.columns.tolist())

# Step 2: Rename columns for consistency if necessary
# Assuming orders has 'CID' and 'VID' representing customer and vendor IDs
if 'CID' in orders.columns:
    orders.rename(columns={'CID': 'customer_id'}, inplace=True)
if 'VID' in orders.columns:
    orders.rename(columns={'VID': 'vendor_id'}, inplace=True)
if 'CID' in customers.columns:
     customers.rename(columns={'CID': 'customer_id'}, inplace=True)
if 'VID' in restaurants.columns:
    restaurants.rename(columns={'VID': 'vendor_id'}, inplace=True)
elif 'id' in restaurants.columns:
    restaurants.rename(columns={'id': 'vendor_id'}, inplace=True)
```

```python
# Step 3: Merge orders with customers
train = orders.merge(customers, on="customer_id", how="left")

# Step 4: Merge with restaurants
train = train.merge(restaurants, on="vendor_id", how="left")

# Step 5: Label target as 1 (positive interactions)
train['target'] = 1

# Final check
print("Merged train shape:", train.shape)
print("Sample rows:\n", train.head())
```

```
/tmp/ipykernel_297/1036775261.py:4: DtypeWarning: Columns (15,16,18,19,20) have
mixed types. Specify dtype option on import or set low_memory=False.
  orders = pd.read_csv("orders.csv")

Orders columns: ['order_id', 'customer_id', 'item_count', 'grand_total',
'payment_mode', 'promo_code', 'vendor_discount_amount',
'promo_code_discount_percentage', 'is_favorite', 'is_rated', 'vendor_rating',
'driver_rating', 'deliverydistance', 'preparationtime', 'delivery_time',
'order_accepted_time', 'driver_accepted_time', 'ready_for_pickup_time',
'picked_up_time', 'delivered_time', 'delivery_date', 'vendor_id', 'created_at',
'LOCATION_NUMBER', 'LOCATION_TYPE', 'CID X LOC_NUM X VENDOR']
Customers columns: ['customer_id', 'gender', 'dob', 'status', 'verified',
'language', 'created_at', 'updated_at']
Restaurants columns: ['id', 'authentication_id', 'latitude', 'longitude',
'vendor_category_en', 'vendor_category_id', 'delivery_charge',
'serving_distance', 'is_open', 'OpeningTime', 'OpeningTime2', 'prepration_time',
'commission', 'is_haked_delivering', 'discount_percentage', 'status',
'verified', 'rank', 'language', 'vendor_rating', 'sunday_from_time1',
'sunday_to_time1', 'sunday_from_time2', 'sunday_to_time2', 'monday_from_time1',
'monday_to_time1', 'monday_from_time2', 'monday_to_time2', 'tuesday_from_time1',
'tuesday_to_time1', 'tuesday_from_time2', 'tuesday_to_time2',
'wednesday_from_time1', 'wednesday_to_time1', 'wednesday_from_time2',
'wednesday_to_time2', 'thursday_from_time1', 'thursday_to_time1',
'thursday_from_time2', 'thursday_to_time2', 'friday_from_time1',
'friday_to_time1', 'friday_from_time2', 'friday_to_time2',
'saturday_from_time1', 'saturday_to_time1', 'saturday_from_time2',
'saturday_to_time2', 'primary_tags', 'open_close_flags', 'vendor_tag',
'vendor_tag_name', 'one_click_vendor', 'country_id', 'city_id', 'created_at',
'updated_at', 'device_type', 'display_orders']

Merged train shape: (135502, 92)
Sample rows:
    order_id customer_id  item_count  grand_total  payment_mode promo_code  \
0  163923.0     KL09J9N         6.0         10.1             1        NaN
```

```
1  163924.0     H5LGGFX         3.0          8.4             1        NaN
2  163925.0     CYLZB6T         4.0         15.0             1        NaN
3  163929.0     4YKUKYN         7.0         27.2             1        NaN
4  163930.0     WDNU3OK         1.0          6.5             1        NaN

   vendor_discount_amount  promo_code_discount_percentage is_favorite  \
0                     0.0                             NaN         NaN
1                     0.0                             NaN         NaN
2                     0.0                             NaN         NaN
3                     0.0                             NaN         NaN
4                     0.0                             NaN         NaN

  is_rated  …                 vendor_tag  \
0       No  …                5,30,48,23
1       No  …         15,34,4,28,27,24,8
2       No  …  2,4,5,8,91,22,12,24,16,23
3       No  …           31,8,10,33,67,21
4       No  …                 1,5,48,16

                                  vendor_tag_name  one_click_vendor  \
0                  Burgers,Fries,Kids meal,Shawarma                 Y
1  Pizzas,Italian,Breakfast,Soups,Pasta,Salads,De…                 Y
2  Arabic,Breakfast,Burgers,Desserts,Free Deliver…                 Y
3     Biryani,Desserts,Indian,Rice,Thali,Vegetarian                 Y
4             American,Burgers,Kids meal,Sandwiches                 Y

   country_id city_id       created_at     updated_at_y device_type  \
0           1       1  9/16/2023 19:37   4/7/2025 21:08           3
1           1       1  8/26/2023 21:47  3/31/2025 22:16           3
2           1       1  1/30/2023 14:42   4/7/2025 15:12           3
3           1       1  1/19/2024 14:01   4/7/2025 20:03           3
4           1       1  1/28/2024 20:37   4/3/2025 22:36           3

  display_orders target
0              1      1
1              1      1
2              1      1
3              1      1
4              1      1

[5 rows x 92 columns]
```

[0]:

[1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# ---------------------------
# Step 0: Create sample data
# ---------------------------
orders = pd.DataFrame({
    'CID': [1, 2, 3, 4, 5],
    'VID': [101, 102, 103, 101, 102]
})

customers = pd.DataFrame({
    'CID': [1, 2, 3, 4, 5],
    'customer_id': ['C1', 'C2', 'C3', 'C4', 'C5']
})

restaurants = pd.DataFrame({
    'VID': [101, 102, 103, 104, 105],
    'vendor_id': ['R1', 'R2', 'R3', 'R4', 'R5']
})
# ---------------------------
# Step 1: Merge all combinations
# ---------------------------
all_customer_vendor = pd.merge(customers, restaurants, how='cross')

# Merge to label existing orders
all_customer_vendor = all_customer_vendor.merge(orders[['CID', 'VID']],␣
 ↪on=['CID', 'VID'], how='left', indicator=True)
all_customer_vendor['target'] = np.where(all_customer_vendor['_merge'] ==␣
 ↪'both', 1, 0)
all_customer_vendor.drop(columns=['_merge'], inplace=True)

# ---------------------------
# Step 2: Balance dataset
# ---------------------------
positive = all_customer_vendor[all_customer_vendor['target'] == 1]
negative = all_customer_vendor[all_customer_vendor['target'] == 0].
 ↪sample(len(positive), random_state=42)
balanced = pd.concat([positive, negative]).reset_index(drop=True)

# ---------------------------
# Step 3: Encode IDs
# ---------------------------
le_cid = LabelEncoder()
le_vid = LabelEncoder()
```

```
balanced['CID'] = le_cid.fit_transform(balanced['customer_id'])
balanced['VID'] = le_vid.fit_transform(balanced['vendor_id'])
# ---------------------------
# Step 4: Train Model
# ---------------------------
features = ['CID', 'VID']
X = balanced[features]
y = balanced['target']

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,␣
 ↪stratify=y, random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# ---------------------------
# Step 5: Evaluation
# ---------------------------
y_pred = model.predict(X_val)
print("Classification Report:\n")
print(classification_report(y_val, y_pred))
```

```
Classification Report:

              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
```

```
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

[2]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import pandas as pd
from sklearn.preprocessing import LabelEncoder

import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Step 1: Load data
orders = pd.read_csv("orders.csv", low_memory=False)
locations = pd.read_csv("train_locations.csv", low_memory=False)

# Step 2: Rename columns for consistency
orders.rename(columns={
    'customer_id': 'CID',
    'vendor_id': 'VID',
    'LOCATION_NUMBER': 'LOC_NUM'
}, inplace=True)

locations.rename(columns={
    'customer_id': 'CID',
    'location_number': 'LOC_NUM'
}, inplace=True)

# Step 3: Merge orders and locations using CID and LOC_NUM
if 'CID' not in orders.columns or 'LOC_NUM' not in orders.columns:
    raise KeyError("Missing CID or LOC_NUM in orders data.")
if 'CID' not in locations.columns or 'LOC_NUM' not in locations.columns:
    raise KeyError("Missing CID or LOC_NUM in locations data.")

merged = pd.merge(orders, locations, on=['CID', 'LOC_NUM'], how='left')

# Step 4: Fill missing target if exists
if 'target' in merged.columns:
    merged['target'] = merged['target'].fillna(0)

# Step 5: Encode CID and VID
le_cid = LabelEncoder()
le_vid = LabelEncoder()

merged['CID_enc'] = le_cid.fit_transform(merged['CID'])
merged['VID_enc'] = le_vid.fit_transform(merged['VID'])

# Step 6: Check sample of processed data
print(merged[['CID', 'VID', 'CID_enc', 'VID_enc']].head())
```

```
       CID  VID  CID_enc  VID_enc
0  KL09J9N   84    15641       18
1  H5LGGFX   78    13002       13
2  CYLZB6T    4     9867        0
3  4YKUKYN  157     3754       34
4  WDNU30K  160    24689       36
```

[3]:
```
# Define the submission DataFrame with selected columns
submission = merged[['CID', 'LOC_NUM', 'VID']].copy()


# Optional: rename columns if required by competition format
# For example, if the submission needs 'customer_id' instead of 'CID':
submission.rename(columns={
    'CID': 'customer_id',
    'LOC_NUM': 'location_number',
    'VID': 'vendor_id'
}, inplace=True)

# Save to CSV
submission.to_csv("Final_Submission.csv", index=False)
print("  Submission file saved as Final_Submission.csv")
print("Columns in merged:", merged.columns.tolist())
```

```
  Submission file saved as Final_Submission.csv
Columns in merged: ['order_id', 'CID', 'item_count', 'grand_total',
'payment_mode', 'promo_code', 'vendor_discount_amount',
'promo_code_discount_percentage', 'is_favorite', 'is_rated', 'vendor_rating',
'driver_rating', 'deliverydistance', 'preparationtime', 'delivery_time',
'order_accepted_time', 'driver_accepted_time', 'ready_for_pickup_time',
'picked_up_time', 'delivered_time', 'delivery_date', 'VID', 'created_at',
'LOC_NUM', 'LOCATION_TYPE', 'CID X LOC_NUM X VENDOR', 'location_type',
'latitude', 'longitude', 'CID_enc', 'VID_enc']
```

[4]:
```
submission.to_csv("Final_Submission.csv", index=False)
print("Submission file saved as Final_Submission.csv")
```

```
Submission file saved as Final_Submission.csv
```