A Report on

Customer Segmentation and Prediction using RFM Analysis and Machine Learning

Submitted by

Ashutosh Sharma

(23070243003)

MSc. Data Science and Spatial Analytics

Under the supervision of

Dr. Rajesh Dhumal

Professor

(Symbiosis Institute of Geoinformatics, Pune)

Batch 2023-2025

Abstract

This project employs RFM (Recency, Frequency, Monetary) analysis along with machine learning techniques to segment customers based on their purchase behavior and predict customer clusters.

The aim is to assist businesses in understanding their customers better, identifying loyal customers, at-risk customers, and other segments, and making data-driven decisions to enhance customer satisfaction and profitability.

Table of Contents

1: Introduction

1.1 Background and Context

The project addresses the need for businesses to understand customer behavior more deeply to tailor marketing strategies effectively. By analyzing RFM scores and using machine learning models, we aim to segment customers into meaningful clusters and predict future customer behavior.

1.2 Problem Statement

The retail industry faces challenges in understanding and segmenting customers based on their purchase behavior. Traditional methods often lack precision and fail to provide actionable insights for targeted marketing strategies.

1.3 Objectives of the Project

- Perform RFM analysis to calculate Recency, Frequency, and Monetary scores for customers.
- Segment customers into clusters based on RFM scores using K-means clustering.
- Train a machine learning model (Random Forest Classifier) to predict customer clusters.
- Provide a user interface for making single customer predictions based on RFM scores.

2: Literature Review

2.1 RFM Analysis

RFM (Recency, Frequency, Monetary) analysis is a method used to evaluate customer value based on their transaction history. It segments customers into categories based on their recent purchases, purchase frequency, and monetary value.

2.2 Customer Segmentation

Customer segmentation involves grouping customers with similar characteristics to enable targeted marketing strategies. Techniques like K-means clustering are commonly used for segmentation based on RFM scores.

2.3 Machine Learning Techniques in Customer Analysis

Machine learning models, such as Random Forest Classifier, are utilized for predicting customer clusters. These models analyze historical data to identify patterns and make predictions about future customer behavior. Methodology

3: Methodology

3.1 Data Collection and Preprocessing

The Online Retail dataset is collected and preprocessed using Python libraries like Pandas. Data cleaning, feature engineering, and Total Amount calculation are performed.

3.2 RFM Score Calculation

Recency, Frequency, and Monetary scores are calculated for each customer based on their purchase history. The scores are then normalized using quantiles.

3.3 RFM Score Normalization

RFM scores are normalized to ensure uniformity and comparability across different customers. Quantiles are used to divide customers into quartiles.

3.4 K-means Clustering

K-means clustering is applied to segment customers into clusters based on their RFM scores. The optimal number of clusters is determined using techniques like the elbow method.

3.5 Model Training and Evaluation

A Random Forest Classifier is trained using RFM scores as features and customer clusters as labels. The model is evaluated using test data to assess its accuracy and performance.

3.6 User Interface Development

A user-friendly interface is developed to accept RFM scores from users and predict the customer cluster using the trained machine learning model.

4: Description of Datasets

4.1 Online Retail Dataset

The dataset contains information about customer transactions, including InvoiceDate, Quantity, UnitPrice, and CustomerID. It serves as the primary data source for RFM analysis.

4.2 RFM Scores Dataset

After RFM analysis and clustering, a dataset containing RFM scores, cluster labels, and other relevant information is created. This dataset is used for model training and evaluation.

5: Results and Discussion

5.1 RFM Analysis Results

Recency, Frequency, and Monetary scores are calculated for each customer, providing insights into their purchase behavior. Histograms and descriptive statistics are used to analyze the distribution of RFM scores.

5.2 Customer Segmentation Results

Customers are segmented into clusters such as Loyal Customers, At-Risk Customers, Elite Customers, etc., based on their RFM scores. Cluster characteristics and names are assigned for interpretation.

5.3 Model Training and Evaluation Results

The Random Forest Classifier achieves a certain accuracy in predicting customer clusters, enabling businesses to make data-driven decisions for targeted marketing campaigns.

5.4 User Interface Functionality

The user interface allows users to input RFM scores and receive predictions about customer clusters. It enhances user experience and facilitates personalized marketing strategies.

6: Analysis of Findings

6.1 Insights from Customer Segmentation

Customer segmentation provides valuable insights into customer behavior and preferences. It helps businesses identify high-value customers, retain loyal customers, and re-engage with at-risk customers.

6.2 Model Performance and Accuracy

The machine learning model demonstrates satisfactory performance in predicting customer clusters. However, continuous evaluation and refinement are necessary to improve accuracy.

6.3 User Feedback and Usability

User feedback on the interface usability and functionality is collected to make improvements and enhance user satisfaction. Iterative testing and updates are conducted based on user input.

7: Conclusion

The project successfully demonstrates the application of RFM analysis and machine learning techniques in customer segmentation and prediction. It provides actionable insights for businesses to improve customer engagement, retention, and profitability.

8: Future Work

Future work may involve:

- Incorporating additional features for improved clustering and prediction.
- Exploring advanced machine learning algorithms for enhanced accuracy.
- Conducting A/B testing and analyzing campaign effectiveness based on predicted customer clusters.

9: References

- Dataset: https://archive.ics.uci.edu/dataset/352/online+retail
- Pandas Documentation: https://pandas.pydata.org/docs/user_guide/index.html#guides
- Scikit-learn Documentation: https://scikit-learn.org/stable/

Appendices

- A. Code Snippets

RFM_Analysis.py | KMeans_Model.py | RandomForest_Model.py | predictor.py

```python
import pandas as pd
import datetime as dt

RFMScores = pd.read_csv('Online_Retail_Train.csv', encoding='unicode_escape')

RFMScores['TotalAmount']=RFMScores['Quantity']*RFMScores['UnitPrice']

RFMScores['InvoiceDate']=pd.to_datetime(RFMScores['InvoiceDate'],dayfirst=True)
Latest_Date = dt.datetime(2011,12,11)
RFMScores = RFMScores.groupby('CustomerID').agg({'InvoiceDate': lambda x: (Latest_Date - x.max()).days, 'InvoiceNo': lambda x: len(x), 'TotalAmount': lambda x: x.sum()})
RFMScores['InvoiceDate'] = RFMScores['InvoiceDate'].astype(int)
RFMScores.rename(columns={'InvoiceDate': 'Recency',
                          'InvoiceNo': 'Frequency',
                          'TotalAmount': 'Monetary'}, inplace=True)

quantiles = RFMScores.quantile(q=[0.25,0.5,0.75])
quantiles = quantiles.to_dict()
def RScoring(x,p,d):
    if x <= d[p][0.25]:
        return 4
    elif x <= d[p][0.50]:
        return 3
    elif x <= d[p][0.75]:
        return 2
    else:
        return 1
def FnMScoring(x,p,d):
    if x <= d[p][0.25]:
        return 1
    elif x <= d[p][0.50]:
        return 2
    elif x <= d[p][0.75]:
        return 3
    else:
        return 4
RFMScores['R'] = RFMScores['Recency'].apply(RScoring, args=('Recency',quantiles,))
RFMScores['F'] = RFMScores['Frequency'].apply(FnMScoring, args=('Frequency',quantiles,))
RFMScores['M'] = RFMScores['Monetary'].apply(FnMScoring, args=('Monetary',quantiles,))
RFMScores['RFMScore'] = RFMScores['R']+RFMScores['F']+RFMScores['M']

Loyalty_Level = ['Bronze', 'Silver', 'Gold', 'Platinum']
Score_cuts = pd.qcut(RFMScores.RFMScore, q = 4, labels = Loyalty_Level)
RFMScores['RFM_Loyalty_Level'] = Score_cuts.values

RFMScores.to_csv('RFMScores.csv', index=False)
```

RFM_Analysis.py | KMeans_Model.py | RandomForest_Model.py | predictor.py

```python
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

RFMScores = pd.read_csv('RFMScores.csv', encoding='unicode_escape')

scaler = StandardScaler()
KMeansRFM = RFMScores.drop(columns=['RFM_Loyalty_Level'])
Scaled_Data = scaler.fit_transform(KMeansRFM)
KMean_clust = KMeans(n_clusters=10, init='k-means++', max_iter=1000, random_state=42, n_init=10)
KMean_clust.fit(Scaled_Data)
RFMScores['Cluster'] = KMean_clust.labels_

cluster_averages = RFMScores.groupby('Cluster').agg({'R': 'mean', 'F': 'mean', 'M': 'mean'})
def assign_cluster_name(row):
    if row['F'] >= 2.5 and row['R'] >= 2.5 and row['M'] < 2.5:
        return 'Loyal Customers'
    elif row['R'] < 2 and row['F'] >= 3 and row['M'] >= 3:
        return 'At-Risk Customers'
    elif row['R'] >= 1.5 and row['F'] >= 1.5 and row['M'] >= 3 and row['R'] < row['M']:
        return 'Elite Customers'
    elif row['R'] > 3 and row['F'] > 3 and row['M'] > 3:
        return 'High-Value Customers'
    elif row['R'] > 3 and row['F'] < 2:
        return 'New Customers'
    elif row['R'] < 2:
        return 'Churned Customers'
    else:
        return 'Undefined'
cluster_averages['Cluster_Name'] = cluster_averages.apply(assign_cluster_name, axis=1)
RFMScores = RFMScores.merge(cluster_averages, left_on='Cluster', right_index=True, how='left')
RFMScores.rename(columns={'R_x': 'R',
                          'F_x': 'F',
                          'M_x': 'M'}, inplace=True)
RFMScores.rename(columns={'R_y': 'R_C.Avg',
                          'F_y': 'F_C.Avg',
                          'M_y': 'M_C.Avg'}, inplace=True)

RFMScores['Cluster'] = RFMScores['Cluster_Name'].factorize()[0]

cluster_counts = RFMScores['Cluster'].value_counts()
clusters_to_keep = cluster_counts[cluster_counts > 1000].index.tolist()
RFMScores = RFMScores[RFMScores['Cluster'].isin(clusters_to_keep)]

unique_clusters = sorted(RFMScores['Cluster'].unique())
cluster_mapping = {cluster_label: idx for idx, cluster_label in enumerate(unique_clusters)}
RFMScores['Cluster'] = RFMScores['Cluster'].map(cluster_mapping)

RFMScores.to_csv('RFMScores.csv', index=False)
```

RFM_Analysis.py | KMeans_Model.py | RandomForest_Model.py | predictor.py

```python
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import pickle
import pandas as pd

RFMScores = pd.read_csv('RFMScores.csv', encoding='unicode_escape')

X = RFMScores[['R', 'F', 'M']]
y = RFMScores['Cluster']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train_scaled, y_train)
y_pred_rf = rf_classifier.predict(X_test_scaled)

model_filename = 'rf.pkl'
with open(model_filename, 'wb') as file:
    pickle.dump(rf_classifier, file)
```

RFM_Analysis.py | KMeans_Model.py | RandomForest_Model.py | predictor.py

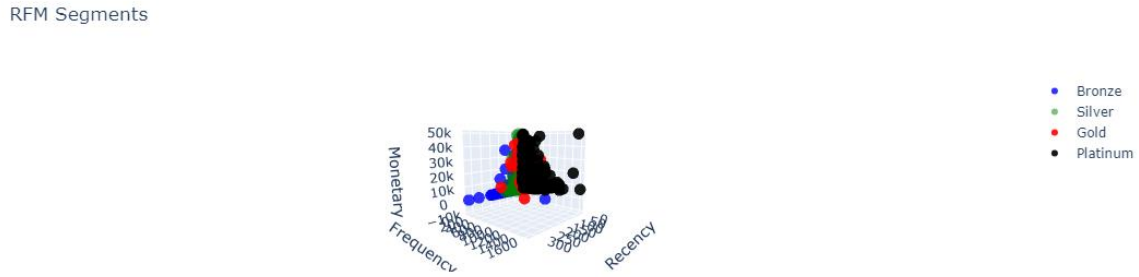
```python
import os
import pickle
import pandas as pd
import RFM_Analysis
import KMeans_Model
import RandomForest_Model
import warnings

warnings.filterwarnings("ignore", category=UserWarning)

def find_pkl_files(directory):
    pkl_files = []
    for filename in os.listdir(directory):
        if filename.endswith('.pkl'):
            pkl_files.append(filename)
    return pkl_files

def load_model_from_file(filename):
    try:
        with open(filename, 'rb') as file:
            loaded_model = pickle.load(file)
        return loaded_model
    except FileNotFoundError:
        print(f"Error: File {filename} not found.")
        return None
    except Exception as e:
        print(f"Error loading model from {filename}: {e}")
        return None

def make_single_prediction(model):
    try:
        recency_value = int(input('Enter Recency: '))
        frequency_value = int(input('Enter Frequency: '))
        monetary_value = int(input('Enter Monetary: '))

        new_data=[[recency_value, frequency_value, monetary_value]]
        if model_filename == 'rf.pkl':
            scaled_new_data = RandomForest_Model.scaler.transform(new_data)

        prediction = model.predict(scaled_new_data)
        return prediction
    except ValueError:
        print("Error: Invalid input. Please enter a valid number.")
        return None
    except KeyboardInterrupt:
        print("\nUser interrupted input.")
        return None
    except EOFError:
        print("\nInput stream closed.")
        return None

directory = '.'
pkl_files = find_pkl_files(directory)
if not pkl_files:
    print("No .pkl files found in the directory.")
else:
    print("Found the following .pkl files in the directory:")
    for file in pkl_files:
        print(file)

    model_filename = input("Enter the name of the .pkl file you want to load: ")
    if model_filename in pkl_files:
        loaded_model = load_model_from_file(model_filename)
        if loaded_model is not None:
            prediction = make_single_prediction(loaded_model)
            if prediction is not None:
                RFMScores = pd.read_csv('RFMScores.csv', encoding='unicode_escape')
                filtered_df = RFMScores[RFMScores['Cluster'].astype(int) == prediction[0]]
                if not filtered_df.empty:
                    print("Prediction: ", filtered_df['Cluster_Name'].iloc[0])
                else:
                    print("Error: Prediction not found in RFMScores.csv.")
            else:
                print("Error: Unable to make prediction.")
    else:
        print(f"Error: {model_filename} not found in the directory.")

```

```
In [13]: runfile('D:/Main/SD/Projects/Python/MachineLearning/Mine/RFMCluster_Prediction_Models/predictor.py', wdir='D:/Main/SD/Projects/Python/MachineLearning/Mine/RFMCluster_Prediction_Models')
Reloaded modules: RFM_Analysis, KMeans_Model, RandomForest_Model
Found the following .pkl files in the directory:
rf.pkl
Enter the name of the .pkl file you want to load: rf.pkl
Enter Recency: 4
Enter Frequency: 1
Enter Monetary: 2
Prediction:  New Customers
```

- B. Additional Charts and Graphs

```
import plotly.graph_objects as go
import pandas as pd

RFMScores = pd.read_csv('RFMScores.csv', encoding='unicode_escape')

graph = RFMScores.query("Monetary < 50000 and Frequency < 2000")

traces = []
for loyalty_level in ['Bronze', 'Silver', 'Gold', 'Platinum']:
    trace = go.Scatter3d(
        x=graph.query(f"RFM_Loyalty_Level ==
'{loyalty_level}'")['Recency'],
        y=graph.query(f"RFM_Loyalty_Level ==
'{loyalty_level}'")['Frequency'],
        z=graph.query(f"RFM_Loyalty_Level ==
'{loyalty_level}'")['Monetary'],
        mode='markers',
        name=loyalty_level,
        marker=dict(
            size=6,
            color='blue' if loyalty_level == 'Bronze' else 'green' if
loyalty_level == 'Silver' else 'red' if loyalty_level == 'Gold' else 'black',
            opacity=0.8 if loyalty_level == 'Bronze' else 0.5 if loyalty_level
== 'Silver' else 0.9 if loyalty_level == 'Gold' else 0.9
        )
    )
    traces.append(trace)

layout = go.Layout(
    scene=dict(
        xaxis=dict(title='Recency'),
        yaxis=dict(title='Frequency'),
        zaxis=dict(title='Monetary'),
    ),
    title='RFM Segments'
)

fig = go.Figure(data=traces, layout=layout)

fig.show()
```

Output:




```
import plotly.graph_objects as go
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import plotly.io as pio
import os
import warnings
from sklearn.exceptions import ConvergenceWarning

warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=ConvergenceWarning)

os.environ['OMP_NUM_THREADS'] = '1'

np.random.seed(0)
X = np.random.rand(100, 2)

scaler = StandardScaler()
Scaled_Data = scaler.fit_transform(X)

sum_of_sq_dist = {}
for k in range(1, 15):
    km = KMeans(n_clusters=k, init='k-means++', max_iter=1000,
n_init=10)
    km = km.fit(Scaled_Data)
    sum_of_sq_dist[k] = km.inertia_

x_vals = list(sum_of_sq_dist.keys())
```

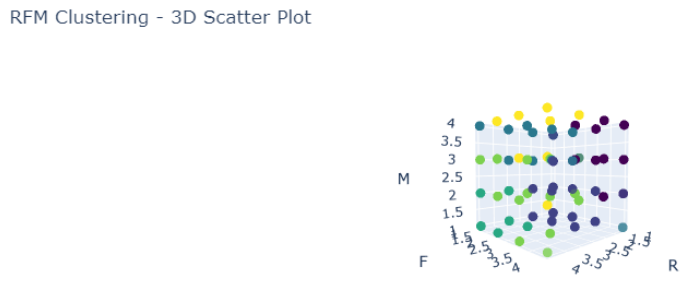
```
y_vals = list(sum_of_sq_dist.values())

fig = go.Figure(data=go.Scatter(x=x_vals, y=y_vals,
mode='lines+markers'))

fig.update_layout(
    xaxis_title='Number of Clusters (k)',
    yaxis_title='Sum of Square Distances',
    title='Elbow Method For Optimal k'
)

pio.show(fig)
```

Output:


```
import plotly.graph_objects as go
import pandas as pd

fig = go.Figure(data=[go.Scatter3d(
    x=RFMScores['R'],
    y=RFMScores['F'],
    z=RFMScores['M'],
    mode='markers',
    marker=dict(
        size=5,
        color=RFMScores['Cluster'],
        colorscale='Viridis',
        opacity=0.8
    )
)])
```

```
fig.update_layout(
    title='RFM Clustering - 3D Scatter Plot',
    scene=dict(
        xaxis_title='R',
        yaxis_title='F',
        zaxis_title='M'
    )
)

fig.show()
```

Output:




```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import RandomForest_Model

cm_rf = confusion_matrix(RandomForest_Model.y_test,
RandomForest_Model.y_pred_rf)

plt.figure(figsize=(8, 6))
plt.imshow(cm_rf, interpolation='nearest', cmap='Blues')
plt.title('Confusion Matrix - Random Forest Classifier')
plt.colorbar()

tick_marks = range(len(cm_rf))
plt.xticks(tick_marks, tick_marks)
plt.yticks(tick_marks, tick_marks)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
```

```
for i in range(len(cm_rf)):
    for j in range(len(cm_rf)):
        plt.text(j, i, str(cm_rf[i, j]), ha='center', va='center', color='black')
plt.tight_layout()

plt.savefig('confusion_matrix_rf.png')
plt.show()
```

Output: