



● Applied Machine Learning Project

LOYAL CUSTOMER SEGMENTATION

Section DBL - DSB

Prepared By

Rodaina Bin Mahfouz	2110271
Joud Faidah	2114874
Muntaha Aldhahri	2110124
Asma Habadi	2112087

Presented To

Dr. Israa Alghanmi





TABLE OF CONTENT

PAGE

Introduction	3
Questions & Hypothesis	4
Tools	5
Tools Definition	6
Data Analysis	7
Results Description	15
Problems	15
Machine Learning Responsibility's	16
Conclusion	17
References	18

Introduction

A company's sales dropped. The board of directors wanted to know their loyal customers so they can reward those loyal customers. The firm management decides to target loyal clients and make some sales offers. The manager sent the collected data and asked for a model be that can be made to predict loyal customers while evaluating the customer data.

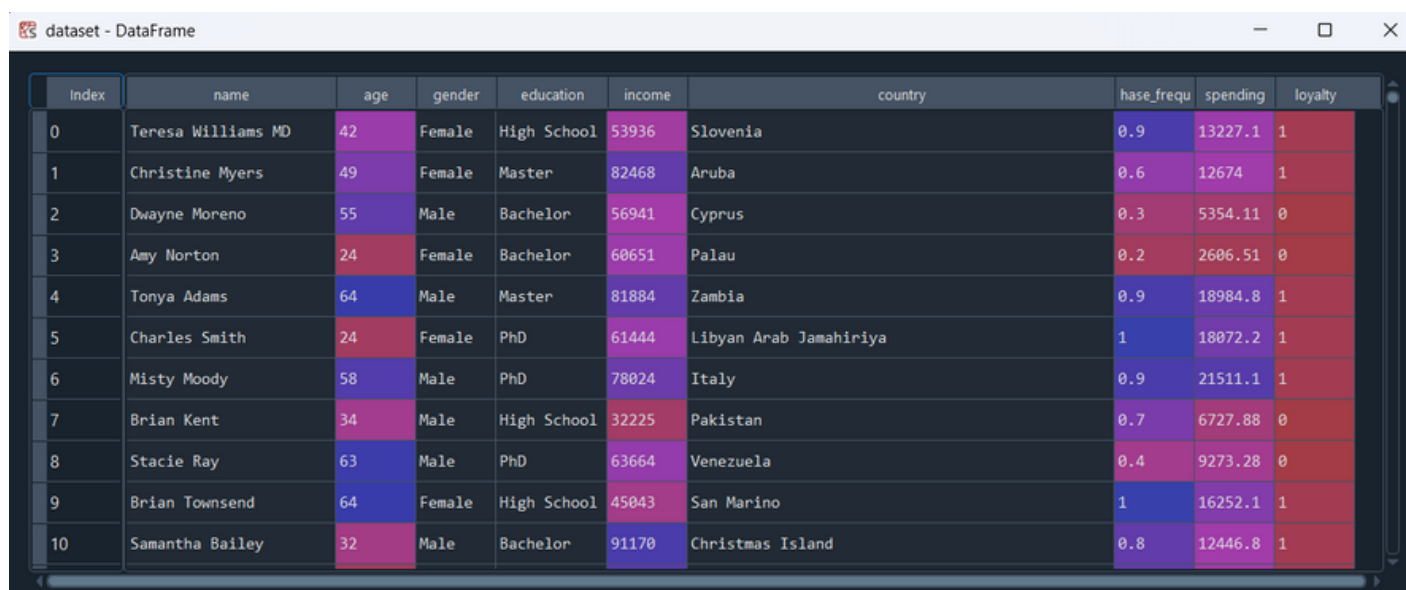
Our goal is to study and compare customer information based on the customers spending, income, and purchase frequency. However, try to classify the loyal customers in the company's data based on customers information's.

Data set Description:

The CSV file provides client information including demographics, income, spending patterns, and purchasing behavior. Each row in the file represents a unique consumer, and the columns indicate various features.

Difficulties:

the data set was a little bit difficult to understand and analyze. To reach the goal and the decision of the Council, we decided to consider customers loyalty based on their purchase frequency if it was more than 6 vest in Baroud of 6 months then it is loyal customer.

A screenshot of a Jupyter Notebook window titled 'dataset - DataFrame'. It displays a DataFrame with 11 columns: Index, name, age, gender, education, income, country, hase_frequ, spending, and loyalty. The data is presented in a table with alternating light and dark blue rows. The 'loyalty' column has values 1 or 0, indicating loyal or non-loyal customers respectively. The 'hase_frequ' column has values 0.2, 0.3, 0.4, 0.6, 0.7, 0.8, 0.9, and 1. The 'spending' column has values ranging from 12446.8 to 18984.8. The 'income' column has values ranging from 53936 to 91170. The 'education' column has values: High School, Master, Bachelor, PhD. The 'country' column has values: Slovenia, Aruba, Cyprus, Palau, Zambia, Libyan Arab Jamahiriya, Italy, Pakistan, Venezuela, San Marino, Christmas Island. The 'age' column has values ranging from 24 to 64. The 'gender' column has values: Female, Male. The 'name' column has values: Teresa Williams MD, Christine Myers, Dwayne Moreno, Amy Norton, Tonya Adams, Charles Smith, Misty Moody, Brian Kent, Stacie Ray, Brian Townsend, Samantha Bailey. The 'Index' column has values from 0 to 10.

Index	name	age	gender	education	income	country	hase_frequ	spending	loyalty
0	Teresa Williams MD	42	Female	High School	53936	Slovenia	0.9	13227.1	1
1	Christine Myers	49	Female	Master	82468	Aruba	0.6	12674	1
2	Dwayne Moreno	55	Male	Bachelor	56941	Cyprus	0.3	5354.11	0
3	Amy Norton	24	Female	Bachelor	60651	Palau	0.2	2606.51	0
4	Tonya Adams	64	Male	Master	81884	Zambia	0.9	18984.8	1
5	Charles Smith	24	Female	PhD	61444	Libyan Arab Jamahiriya	1	18072.2	1
6	Misty Moody	58	Male	PhD	78024	Italy	0.9	21511.1	1
7	Brian Kent	34	Male	High School	32225	Pakistan	0.7	6727.88	0
8	Stacie Ray	63	Male	PhD	63664	Venezuela	0.4	9273.28	0
9	Brian Townsend	64	Female	High School	45043	San Marino	1	16252.1	1
10	Samantha Bailey	32	Male	Bachelor	91170	Christmas Island	0.8	12446.8	1



Questions & Hypothesis

Question 1: What role does the predictive model play in the company's strategy, and how is it expected to contribute to the identification and updating of loyal customers?

Question 2: Are there any specific expectations or goals outlined for the study and comparison of customer information within the dataset?

Hypothesis 1: The model will use consumer information such as demographics, income, spending habits, and purchasing habits to identify patterns indicative of loyalty. The prediction model is designed to improve the company's capacity to engage and keep its most valuable clients proactively through customized sales offers.

Hypothesis 2: The company has established particular objectives, such as finding significant demographic characteristics impacting expenditure, discovering correlations between income levels and purchasing behavior, and comprehending the impact of prior interaction on customer loyalty. According to the theory, these insights will influence focused marketing activities and budget allocation.



Tools

1. SVM :

- 1.1 import numpy
- 1.2 import matplotlib.pyplot
- 1.3 import pandas
- 1.4 from sklearn.model_selection import train_test_split
- 1.5 from sklearn.preprocessing import StandardScaler
- 1.6 from sklearn.svm import SVC
- 1.7 from sklearn.metrics import confusion_matrix
- 1.8 from matplotlib.colors import ListedColormap

2. K-NN :

- 2.1 import numpy
- 2.2 import matplotlib.pyplot
- 2.3 import pandas
- 2.4 from sklearn.model_selection import train_test_split
- 2.5 from sklearn.preprocessing import StandardScaler
- 2.6 from sklearn.neighbors import KNeighborsClassifier
- 2.7 from sklearn.metrics import confusion_matrix
- 2.8 from matplotlib.colors import ListedColormap

3. Random Forest :

- 3.1 import numpy
- 3.2 import matplotlib.pyplot
- 3.3 import pandas
- 3.4 from sklearn.model_selection import train_test_split
- 3.5 from sklearn.preprocessing import StandardScaler
- 3.6 from sklearn.ensemble import RandomForestClassifier
- 3.7 from sklearn.metrics import confusion_matrix
- 3.8 from matplotlib.colors import ListedColormap

4. Logistic Regression :

- 4.1 import numpy
- 4.2 import matplotlib.pyplot
- 4.3 import pandas
- 4.4 from sklearn.model_selection import train_test_split
- 4.5 from sklearn.preprocessing import StandardScaler
- 4.6 from sklearn.linear_model import LogisticRegression
- 4.7 from sklearn.metrics import confusion_matrix
- 4.8 from matplotlib.colors import ListedColormap



Tools Definition

NumPy: This imports the NumPy library, which is commonly used for numerical operations in Python.

Matplotlib.pyplot: This imports the pyplot module from the matplotlib library, which is used for creating visualizations like plots and charts.

Pandas: This imports the Pandas library, a powerful data manipulation and analysis library often used for handling structured data.

train_test_split: This imports the train_test_split function from the model_selection module in the scikit-learn library. It's used for splitting datasets into training and testing sets.

StandardScaler: This imports the StandardScaler class from the preprocessing module in scikit-learn. It's used for standardizing features by removing the mean and scaling to unit variance.

confusion_matrix: This imports the confusion_matrix function from the metrics module in scikit-learn. It's often used to evaluate the performance of a classification algorithm.

ListedColormap: This imports the ListedColormap class from the colors module in Matplotlib. It can be used for customizing colormaps in visualizations.

SVC: This imports the Support Vector Classification (SVC) algorithm from the Support Vector Machines (SVM) module in scikit-learn. SVM is a machine learning algorithm used for classification tasks.

KNeighborsClassifier: This imports the KNeighborsClassifier class from the neighbors module in scikit-learn. It's used for k-nearest neighbors classification.

RandomForestClassifier: This imports the RandomForestClassifier class from the ensemble module in scikit-learn. Random Forest is an ensemble learning method often used for classification tasks.

LogisticRegression: This imports the LogisticRegression class from the linear_model module in scikit-learn. Logistic Regression is a linear model used for binary and multiclass classification.

Data Analysis

1 - Loading Data:

```
#-----  
# Importing the dataset  
dataset = pd.read_csv('customer_data.csv')
```

2 - Independent and dependent variable:

```
# Dependent & Independent:  
# Independent variable(Features) = Demographic information "spending, purchase frequency"  
X = dataset[['purchase_frequency', 'spending']].values  
# Dependent variable(Target) = loyal customer.  
y = dataset['loyalty'].values
```

X - NumPy object array		
	0	1
0	0.9	13227.1
1	0.6	12674
2	0.3	5354.11
3	0.2	2606.51
4	0.9	18984.8
5	1	18872.2
6	0.9	21511.1
7	0.7	6727.88
8	0.4	9273.28
9	1	16252.1
10	0.8	12446.8

y - NumPy object array		
	0	
0	1	
1	1	
2	0	
3	0	
4	1	
5	1	
6	1	
7	0	
8	0	
9	1	
10	1	

3 - Data Preprocessing:

```
#-----  
# Data cleaning  
# replace any missing value in spending with median of spending  
median_spending = np.median(y)  
y = (y > median_spending).astype(int)
```

```
#-----  
# Feature Scaling  
from sklearn.preprocessing import StandardScaler  
sc = StandardScaler()  
X_train = sc.fit_transform(X_train)  
X_test = sc.transform(X_test)
```

4 - Splitting the data into Training and Testing sets:

```
#-----  
# Splitting the dataset into the Training set and Test set  
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Splitting the data into (Training set of 0.75), (Testing set of 0.25), and (Random state of 42)



5 - Applying machine learning models:

Four models have been relied upon to classify, which are as follows:

#1 Support Vector Machine (SVM) --> used for classification and regression tasks. It is particularly effective for high-dimensional spaces and is well-suited for both linear and non-linear datasets.

```
#-----  
# Training the SVM model on the training set  
from sklearn.svm import SVC  
classifier = SVC(kernel='rbf', random_state=42)  
classifier.fit(X_train, y_train)
```

#2 K-Nearest Neighbors (K-NN) --> used for both classification and regression tasks. It's a type of instance-based learning where the algorithm makes predictions based on the majority class or average.

```
#-----  
# Training the K-nn model  
from sklearn.neighbors import KNeighborsClassifier  
classifier = KNeighborsClassifier(n_neighbors=5, p=5)  
classifier.fit(X_train, y_train)
```

#3 Logistic Regression --> used for binary classification problems, where the goal is to predict whether an instance belongs to one of two classes.

```
#-----  
# Training the Logistic Regression model on the Training set  
from sklearn.linear_model import LogisticRegression  
classifier = LogisticRegression(random_state = 42)  
classifier.fit(X_train, y_train)
```

#4 Random Forest --> used for both classification and regression tasks in machine learning. It operates by constructing a multitude of decision trees during training and outputs.

```
#-----  
# Training the Random Forest  
from sklearn.ensemble import RandomForestClassifier  
classifier = RandomForestClassifier(n_estimators=100, random_state=42)  
classifier.fit(X_train, y_train)
```


6 - Confusion matrix & Accuracy of all machines learning algorithms:

Support Vector Machine (SVM)

```
#-----  
# Making the Confusion Matrix  
from sklearn.metrics import confusion_matrix  
cm = confusion_matrix(y_test, y_pred)  
print("\n ***** SVM *****")  
print(f"Classification matrix: \n{cm} \n")  
  
Ac= classifier.score(X_test, y_test)  
print(f"Accuracy: {Ac} \n")
```

```
***** SVM *****  
Classification matrix:  
[[137  6]  
 [  4 103]]  
  
Accuracy: 0.96
```

K-Nearest Neighbors (K-NN)

```
#-----  
# Making the Confusion Matrix  
from sklearn.metrics import confusion_matrix  
cm = confusion_matrix(y_test, y_pred)  
print("\n ***** K-NN *****")  
print(f"Classification matrix: \n{cm} \n")  
  
Ac= classifier.score(X_test, y_test)  
print(f"Accuracy: {Ac} \n")
```

```
***** K-NN *****  
Classification matrix:  
[[134  9]  
 [  4 103]]  
  
Accuracy: 0.948
```

Logistic Regression

```
#-----  
# Making the Confusion Matrix  
from sklearn.metrics import confusion_matrix  
cm = confusion_matrix(y_test, y_pred)  
print("\n ***** Logistic Regression *****")  
print(f"Classification matrix: \n{cm} \n")  
  
Ac= classifier.score(X_test, y_test)  
print(f"Accuracy: {Ac} \n")
```

```
***** Logistic Regression *****  
Classification matrix:  
[[133 10]  
 [  4 103]]  
  
Accuracy: 0.944
```

Random Forest

```
#-----  
# Making the Confusion Matrix  
from sklearn.metrics import confusion_matrix  
cm = confusion_matrix(y_test, y_pred)  
print("\n ***** Random Forest *****")  
print(f"Classification matrix: \n{cm} \n")  
  
Ac= classifier.score(X_test, y_test)  
print(f"Accuracy: {Ac} \n")
```

```
***** Random Forest *****  
Classification matrix:  
[[136  7]  
 [  5 102]]  
  
Accuracy: 0.952
```

Machine Learning Algorithms Accuracy Review.

All the algorithms that have been used to predict loyal customers has shown high performance to reach the goal of the project. But the best algorithm of those four (SVM, K-NN, logistic regression, and Random Forest) is the Support Vector Machine (SVM). The Support Vector Machine (SVM) has the higher accuracy of 0.96.

Machine Learning Algorithms Ranking by Highest Accuracy:

- 1- Support Vector Machine (SVM) = 0.96
- 2- Random Forest = 0.95
- 3- K-Nearest Neighbors (K-NN) = 0.948
- 4- Logistic Regression = 0.944

7 - Prediction the test set results of all machines learning algorithms:

In all four machine learning algorithms (SVM, K-NN, logistic regression, and Random Forest), the prediction of test set results has the same value and code.

```
#-----  
# Predicting the Test set results  
y_pred = classifier.predict(X_test)
```

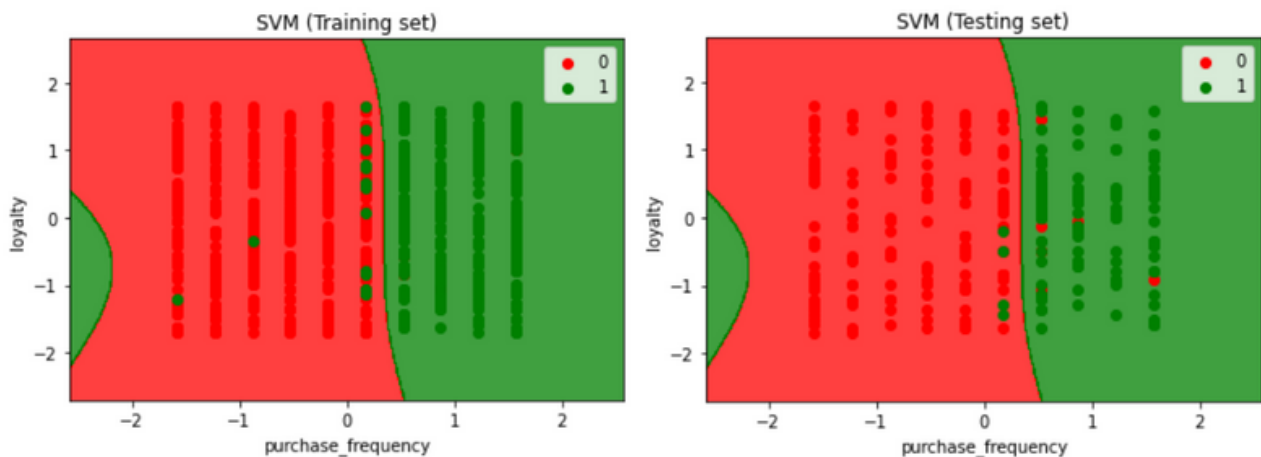
y - NumPy object array

	0
0	1
1	1
2	0
3	0
4	1
5	1
6	1
7	0
8	0
9	1
10	1

8 - Visualizing all machine learning algorithms:

Support Vector Machine (SVM)

```
#-----  
# Visualising sets  
# Visualising the Training set results  
from matplotlib.colors import ListedColormap  
X_set, y_set = X_train, y_train  
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),  
                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))  
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),  
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))  
plt.xlim(X1.min(), X1.max())  
plt.ylim(X2.min(), X2.max())  
for i, j in enumerate(np.unique(y_set)):  
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],  
               c = ListedColormap(('red', 'green'))(i), label = j)  
plt.title('SVM (Training set)')  
plt.xlabel('purchase_frequency')  
plt.ylabel('loyalty')  
plt.legend()  
plt.show()  
  
# Visualising the Test set results  
from matplotlib.colors import ListedColormap  
X_set, y_set = X_test, y_test  
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),  
                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))  
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),  
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))  
plt.xlim(X1.min(), X1.max())  
plt.ylim(X2.min(), X2.max())  
for i, j in enumerate(np.unique(y_set)):  
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],  
               c = ListedColormap(('red', 'green'))(i), label = j)  
plt.title('SVM (Testing set)')  
plt.xlabel('purchase_frequency')  
plt.ylabel('loyalty')  
plt.legend()  
plt.show()
```

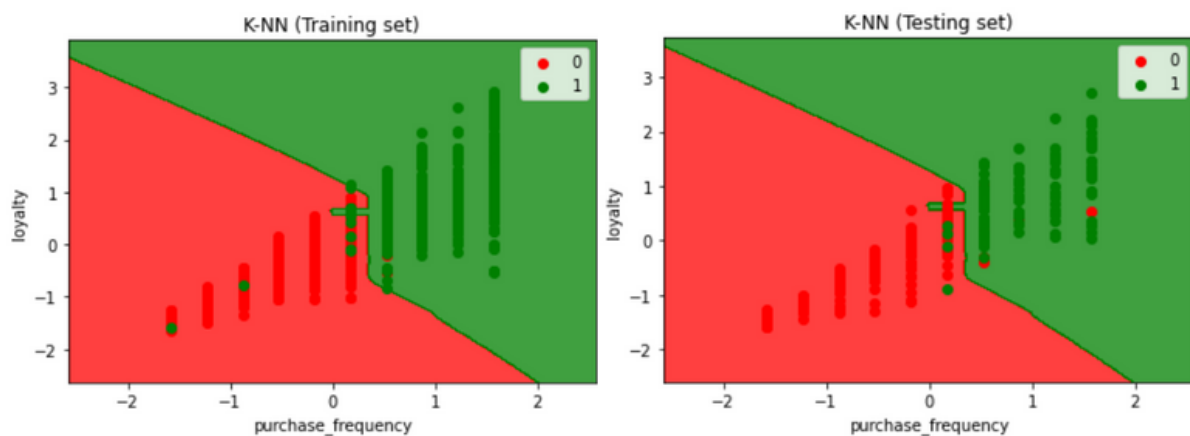


Ac	float64	1	0.96
classifier	svm._classes.SVC	1	SVC object of sklearn.svm._classes module
cm	Array of int64	(2, 2)	<div>[[137 6] [4 103]]</div>

K-Nearest Neighbors (K-NN)

```
#-----
# Visualising sets
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Training set)')
plt.xlabel('purchase_frequency')
plt.ylabel('Loyalty')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Testing set)')
plt.xlabel('purchase_frequency')
plt.ylabel('Loyalty')
plt.legend()
plt.show()
```



Ac	float64	1	0.948
classifier	neighbors._classification.KNeighborsClassifier	1	KNeighborsClassifier ...
cm	Array of int64	(2, 2)	$\begin{bmatrix} 134 & 9 \\ 4 & 103 \end{bmatrix}$

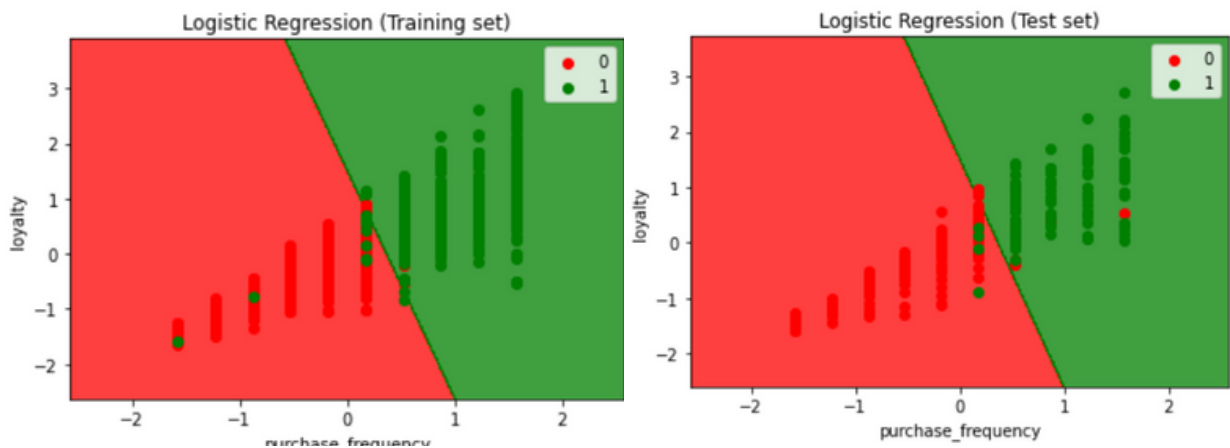
Logistic Regression

```
#-----
# Visualising sets
# Visualising the Training set results
from matplotlib.colors import ListedColormap

X_set, y_set = X_train, y_train
x1, x2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(x1, x2, classifier.predict(np.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(x1.min(), x1.max())
plt.ylim(x2.min(), x2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('purchase_frequency')
plt.ylabel('loyalty')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap

X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('purchase_frequency')
plt.ylabel('loyalty')
plt.legend()
plt.show()
```



Ac	float64	1	0.944
classifier	linear_model.logistic.LogisticRegression	1	LogisticRegression object ...
cm	Array of int64	(2, 2)	[[133 10] [4 103]]

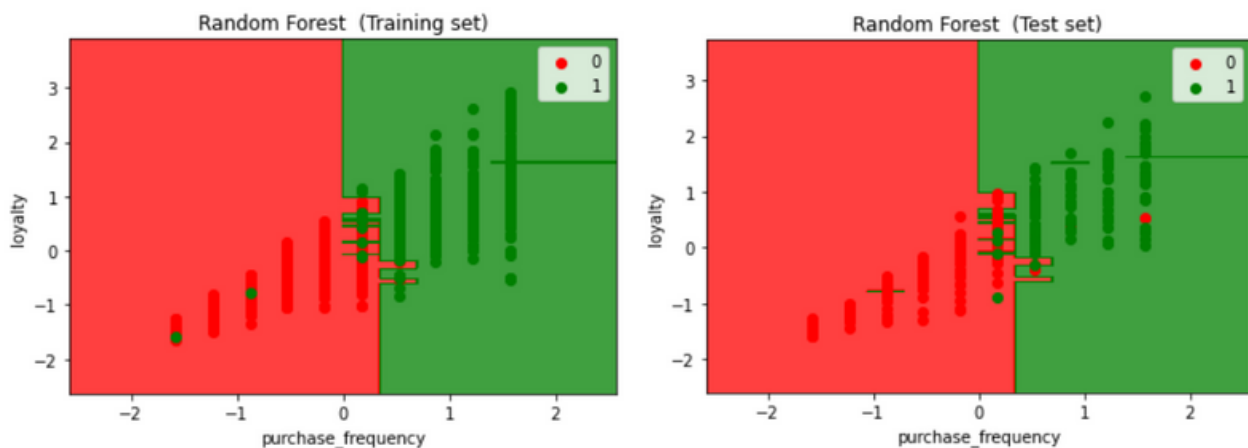
Random Forest

```
#-----
# Visualising sets
# Visualising the Training set results
from matplotlib.colors import ListedColormap

X_set, y_set = X_train, y_train
x1, x2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(x1, x2, classifier.predict(np.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(x1.min(), x1.max())
plt.ylim(x2.min(), x2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Random Forest (Training set)')
plt.xlabel('purchase_frequency')
plt.ylabel('loyalty')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap

X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Random Forest (Test set)')
plt.xlabel('purchase_frequency')
plt.ylabel('loyalty')
plt.legend()
plt.show()
```



Ac	float64	1	0.952
classifier	ensemble._forest.RandomForestClassifier	100	RandomForestClassifier objec...
cm	Array of int64	(2, 2)	[[136 7] [5 102]]



Results Description:

The predictive model is a key part of a company's strategy, utilizing customer data to forecast and identify loyal customers. It dynamically updates itself to adapt to changing customer needs, enhancing customer engagement and retention.

The company is analyzing customer data to identify demographic factors, income correlations, and customer loyalty impact, aiming to refine marketing strategies, enhance customer segmentation, and optimize resource allocation.

Answering Questions:

Answer 1: The predictive model is a key part of a company's strategy, utilizing customer data to forecast and identify loyal customers. It dynamically updates itself to adapt to changing customer needs, enhancing customer engagement and retention.

Answer 2: The company is analyzing customer data to identify demographic factors, income correlations, and customer loyalty impact, aiming to refine marketing strategies, enhance customer segmentation, and optimize resource allocation.

Difficulties During Building the Models:

I encountered numerous programming issues with my team as we worked to accomplish the goal and create a successful working model. One of the most significant is not being aware of the model's best attribute according to Choies. Constructing a visualization presented another challenge.



Conclusion:

The predictive modeling initiative aims to identify and engage loyal customers by analyzing demographic and income data. The project uses machine learning models like SVM, K-NN, Random Forest, and Logistic Regression to segment customers based on factors like purchase frequency and satisfaction. This approach enhances customer satisfaction and fosters a stronger, more profitable relationship with the consumer base.

The Support Vector Machine (SVM) algorithm stands out as the best for predicting customer loyalty, demonstrating the highest accuracy (0.96) and effectively balancing true and false classifications. Its robust performance, particularly in handling diverse data like income, spending patterns, and purchase frequency, makes it the most reliable choice for this task.



Machine Learning Responsibility's:

The ethical and practical issues surrounding the development, execution, and application of machine learning (ML). Data scientists, engineers, ethicists, legislators, and other stakeholders must all be involved in machine learning tasks, which call for a comprehensive and interdisciplinary approach. It's divided into two main type:

- * Ethical Responsibility.
- * Professional Responsibility.

1 - Ethical Responsibility's:

Transparency: A lack of openness in ML models might make it difficult to understand the model. Document the creation process of the ML model, employ interpretable models whenever possible, and provide clear explanations for model predictions.

Privacy: ML models may be exposed to sensitive personal information. Implement strong data protection safeguards, anonymize data whenever possible, and clearly express data usage regulations to consumers.

Fairness and Bias: ML models can inherit biases from training data, and give use discriminatory outcomes. you should assess biases in the data and algorithms.

2 - Professional Responsibility's:

Data Quality: Poor data quality can lead to faulty models. Data should be cleaned and validated. Document data sources and preprocessing methods clearly.

Legal and ethical compliance: Failure to follow laws or ethical standards can have serious implications. You must ensure that all applicable laws and ethical principles are followed.



Referencess:

<https://www.kaggle.com/datasets/goyaladi/customer-spending-dataset/data>

<https://www.w3.org/TR/webmachinelearning-ethics/#ethical-principles-for-web-ml>

<https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/>

<https://www.simplilearn.com/tutorials/scikit-learn-tutorial/sklearn-svm-support-vector-machines#:~:text=It's%20a%20C%2Dbased%20support,one%2Dt%20Done%20mechanism.>

<https://www.educative.io/answers/kneighborsclassifier-in-scikit-learn>

<https://builtin.com/data-science/random-forest-python-deep-dive>

<https://library.virginia.edu/data/articles/logistic-regression-four-ways-with-python#:~:text=Logistic%20regression%20is%20a%20predictive,corresponding%20dependent%20variables%2C%20or%20responses.>