KBTU KAZAKH-BRITISH TECHNICAL UNIVERSITY

REPORT

on Data Mining/

Customer Segmentation and Prediction Using Data Mining in Python

student________________Serikkanov Nurtas Mukhituly_____________________

(surname, first name, patronymic)

year of study______4______specialty / educational program_________________

_______________________Information System ___________________________

_____________School of Information Technologies and Engineering___________

_________________________Date: October 2024_________________________

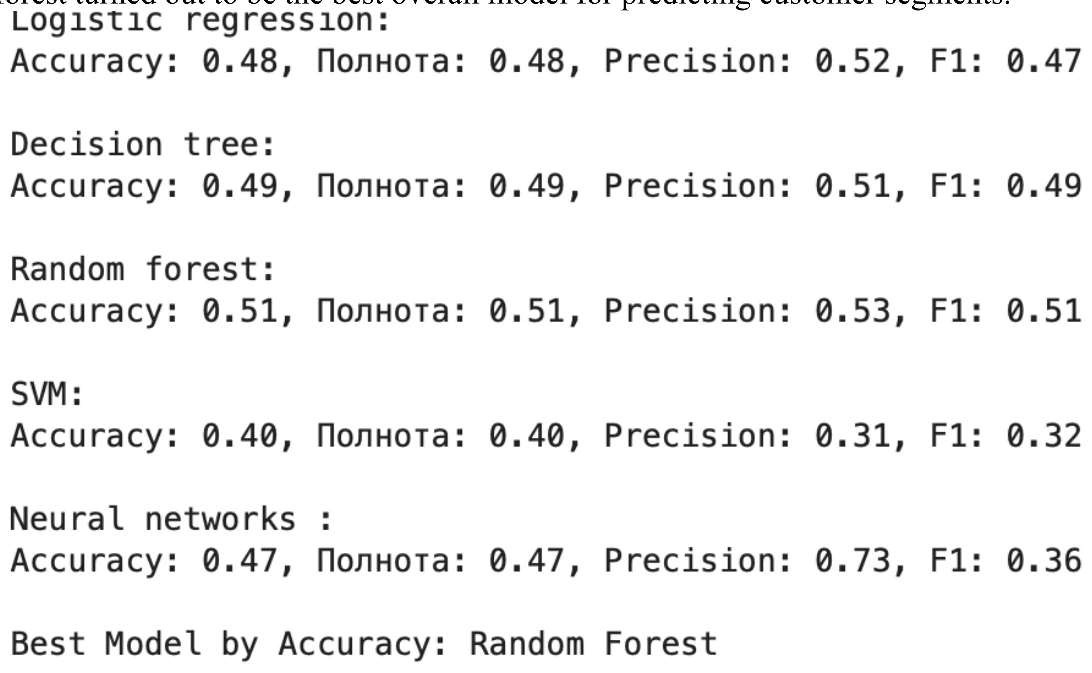
Table of Contents

1. Executive Summary

This project was on customer segmentation and behavior prediction by using various techniques in data mining. It focused on workout data from different customers with the objective of defining different user segments and some classification models for predicting behavior.

Basic and advanced classification methods were used for the determined goals. First of all, logistic regression and decision trees were applied for the first step of customer segments classification, then the advanced methods such as random forest, support vector machine, and neural networks were run in order to improve the results and their results were compared with the basic algorithms.

It had the best result in terms of accuracy, recall, and precision; therefore, the random forest turned out to be the best overall model for predicting customer segments.



```
Logistic regression:
Accuracy: 0.48, Полнота: 0.48, Precision: 0.52, F1: 0.47

Decision tree:
Accuracy: 0.49, Полнота: 0.49, Precision: 0.51, F1: 0.49

Random forest:
Accuracy: 0.51, Полнота: 0.51, Precision: 0.53, F1: 0.51

SVM:
Accuracy: 0.40, Полнота: 0.40, Precision: 0.31, F1: 0.32

Neural networks :
Accuracy: 0.47, Полнота: 0.47, Precision: 0.73, F1: 0.36

Best Model by Accuracy: Random Forest
```

The best performing model subsequently classified customers according to various parameters related to physical activity, and hence useful recommendations may be provided for personalized customer engagement strategies.

It explains the methodology adopted in the project, the challenges faced, and the results obtained.

2. Introduction

Data Mining is a process of extracting useful information from large volumes of data in order to discover hidden patterns, trends, and relationships. It is now being applied in a wide field of industries to arrive at informed decisions based on data. In business terms, this would mean a better understanding of customers' behavior, the targeting of audiences into well-focused groups, and the establishment of personalized marketing strategies for each. Customer segmentation, done in view of data mining techniques, enjoys great prominence in

modern analytics, as it might detect an organization that has groups of customers with diverse needs and preferences. It enhances customer experiences and improves loyalty while optimizing marketing and operational processes.
The motivation for choosing this project is because, as with almost any industry, the fitness industry needs to develop a more intimate understanding of its customer base in order to further improve the services offered. Based on the customer's fitness and activity data, several groups of users can be singled out and their behavior forecast. This would give fitness centers the opportunity to offer personalized training programs and diets, hence improving customer engagement and results.
Also, modern gyms collect a vast amount of data on their customers about parameters such as age, sex, weight, activity level, training frequency, and other relevant characteristics. Without the application of data mining methods, this database is not exploited, and neither does it make a competitive advantage for the business. This will also involve developing models from this data that will contribute in not only the segmentation of clients but also in predicting their future actions regarding either continuing their membership or moving on to more intense training programs.
This, therefore, means that the project will contribute to the fitness centers by enhancing the quality of service, improving customer retention through recommendations given with more precision, and developing customized programs based on data analysis that will give rise to a better overall experience of the customer with the fitness center.

3. Project Objectives

The main objectives of the project are:
1. Segmentation of the customer: As guided by user physical characteristics and activity data, identify different groups of customers presenting similar behavior patterns. This will help management in fitness centers understand in-depth which training programs and services are most sought after by different customer segments.
2. Predicting Customer Behavior: By using machine learning models, I aimed to predict how customers might behave in the future—like whether they'll stick with their membership, adjust their workout intensity, or increase their use of different services.
3. To evaluate the different classification models' performance, I applied logistic regression, decision trees, random forests, and neural networks to predict customers' behavior. Then, I evaluated the performance of each model using the metrics, which included accuracy, and precision, recall, F1 score.
4. Recommendations to Improve Customer Experience: Segmentation of customer behavior also let me make strategic recommendations on the personalization of services to the fitness centers. I supported them with particular ideas on how they could retain more customers, as well as increase overall satisfaction.

4. Data Mining Overview

Data mining refers to the extraction process of data from hitherto ignored large volumes of information. The main aim of Data Mining will be to find out such hidden patterns and trends, which would turn useful in various sectors for decision-making, forecasting, and process optimization. Some of the major concepts with which Data Mining operates are:

- *Classification*: A method that attempts to project categories for objects, given the input data. Examples can include the segregation of customers by risk classes or purchasing categories.

- *Clustering*: This is a process of grouping objects that operate based on similar characteristics; thus, it may enable one to mark segments of customers or highlight trends of behavior.
- *Regression*: This projects numeric values based on variable relationships. This could be revenue or cost forecasting.
- *Association rules*: These include methods for finding frequent relationships between variables. Useful for the analysis of user behavior, such as combined purchases.

Data Mining methodologies include the following:

- CRISP-DM, or Cross-Industry Standard Process for Data Mining, is among the most widely used methodologies. It includes the following phases: business understanding, data understanding, data preparation, modeling, evaluation, and implementation.
- SEMMA's- developed by SAS-approach is: Sample, Explore, Modify, Model, Assess. This terminology describes the process of working with data in a sequence: sampling, research, modification, modeling, and evaluation.

Applications of Data Mining in various sectors:

- *Business*: analysis of consumer behavior, demand forecasting, marketing analytics.
- *Medicine*: diagnosis of a disease, prognosis, personalized medicine.
- *Finance*: risk management, stock price forecasting, fraud detection.
- *Fitness and sport*: personalization of the training process, prediction of customer outcome, improvement of user retention.

Thus, Data Mining is widely applied to enhance process understanding, better decision-making, and efficiency optimization in so many sectors, including the fitness industry, and is relevant for the project.

5. Data Preprocessing

5.1. Data Collection For the purpose of this work, I will be using a simulated dataset representing data for clients in gyms. The key features of client activity and demographics, as shown below, can be viewed in this dataset:

The sources of the data will include, but are not limited to, a registration form, activity tracker, or any other survey that is filled out by clients.

- Dataset size: The dataset is composed of 1,000 clients who have different features describing them.
- Key Features: Some of the most critical features would include the following: age, gender, weight, height, max heart rate, average heart rate, workout duration, calories burned, weekly workout frequency, body fat percentage, water intake, and so on.

```python
# Nurtas Serikkanov
import pandas as pd

data = pd.read_csv('gym_members_exercise_tracking.csv')
print(data.head())
```

```
   Age  Gender  Weight (kg)  Height (m)  Max_BPM  Avg_BPM  Resting_BPM  \
0   56    Male         88.3        1.71      180      157           60   
1   46  Female         74.9        1.53      179      151           66   
2   32  Female         68.1        1.66      167      122           54   
3   25    Male         53.2        1.70      190      164           56   
4   38    Male         46.1        1.79      188      158           68   

   Session_Duration (hours)  Calories_Burned Workout_Type  Fat_Percentage  \
0                      1.69           1313.0         Yoga            12.6   
1                      1.30            883.0         HIIT            33.9   
2                      1.11            677.0       Cardio            33.4   
3                      0.59            532.0     Strength            28.8   
4                      0.64            556.0     Strength            29.2   

   Water_Intake (liters)  Workout_Frequency (days/week)  Experience_Level  \
0                    3.5                              4                 3   
1                    2.1                              4                 2   
2                    2.3                              4                 2   
3                    2.1                              3                 1   
4                    2.8                              3                 1   

     BMI  
0  30.20  
1  32.00  
2  24.71  
3  18.41  
4  14.39  
```

Full list of features:

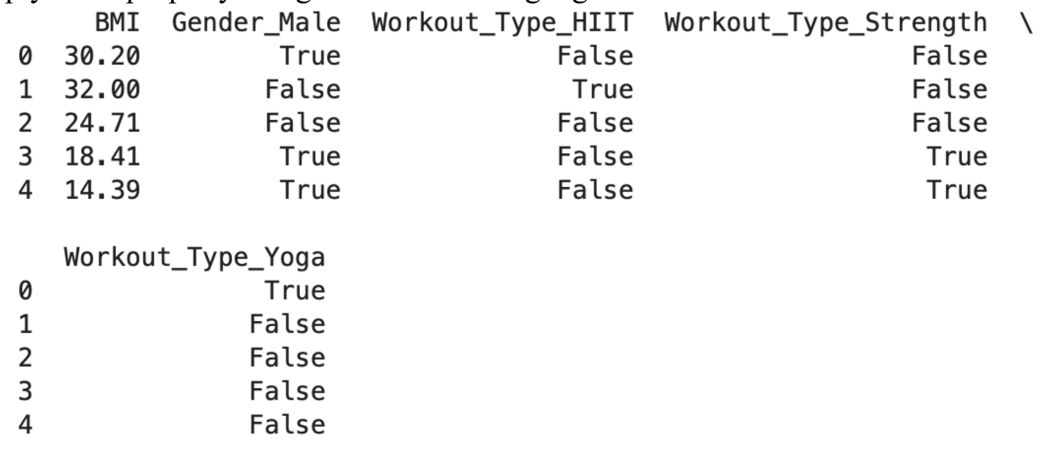
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 973 entries, 0 to 972
Data columns (total 15 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Age                            973 non-null    int64  
 1   Gender                         973 non-null    object 
 2   Weight (kg)                    973 non-null    float64
 3   Height (m)                     973 non-null    float64
 4   Max_BPM                        973 non-null    int64  
 5   Avg_BPM                        973 non-null    int64  
 6   Resting_BPM                    973 non-null    int64  
 7   Session_Duration (hours)       973 non-null    float64
 8   Calories_Burned                973 non-null    float64
 9   Workout_Type                   973 non-null    object 
 10  Fat_Percentage                 973 non-null    float64
 11  Water_Intake (liters)          973 non-null    float64
 12  Workout_Frequency (days/week)  973 non-null    int64  
 13  Experience_Level               973 non-null    int64  
 14  BMI                            973 non-null    float64
dtypes: float64(7), int64(6), object(2)
memory usage: 114.2+ KB
None
```

5.2. Data Cleaning and Preparation To prepare the dataset for analysis, I first had to clean and encode some of the features:

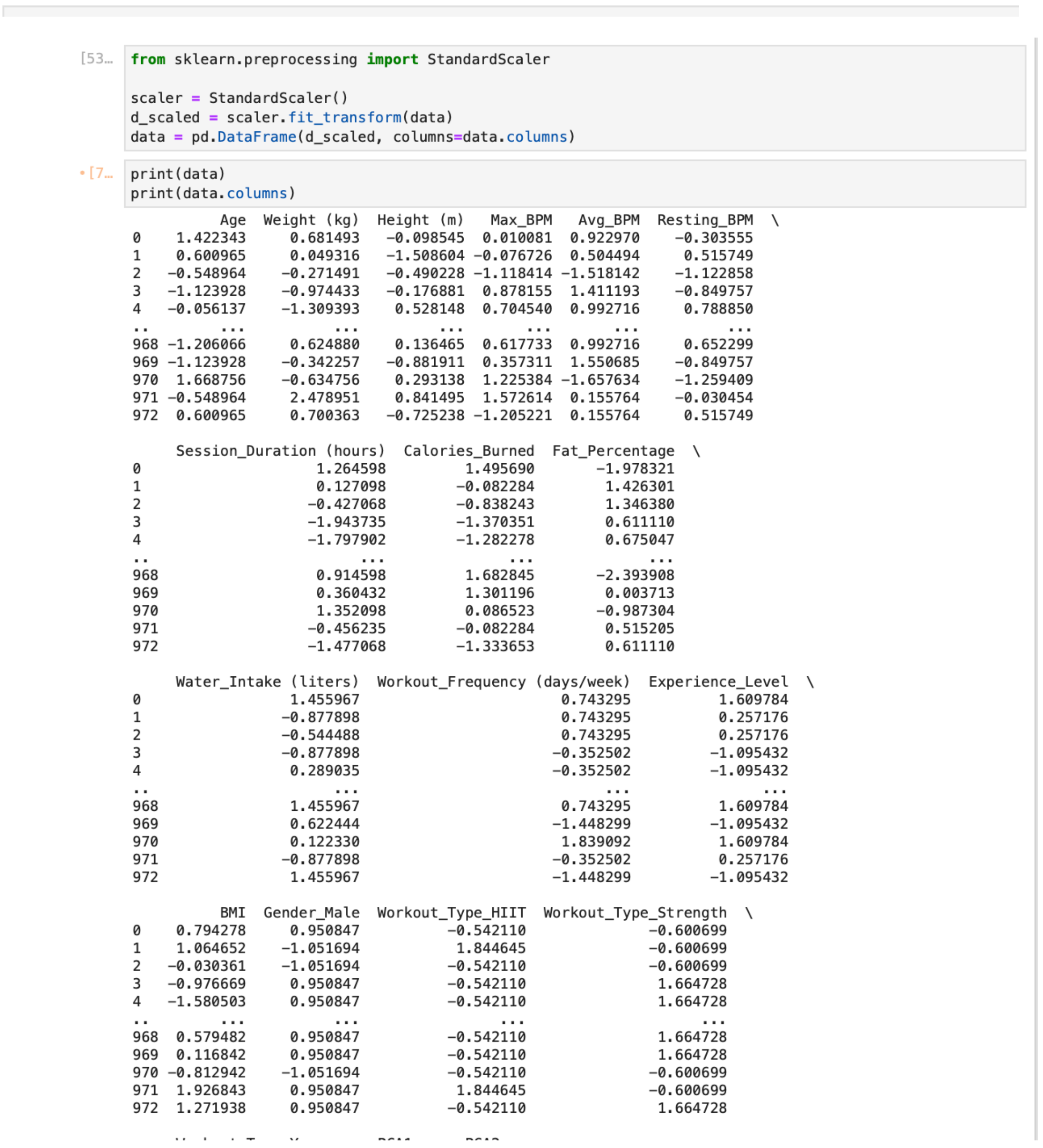
Encoding Categorical Variables: One such feature "Gender" was converted to a binary format such as '0' for male and '1' for female. It made computation easy for machine learning algorithms.

One-Hot Encoding: Other categorical variables were addressed using one-hot encoding. For instance, classes within the "Workout_Type" feature included cardio, strength training, and yoga, and this encoding changed them into a form that these algorithms support. It encoded the feature by representing each category as a different binary feature variable to take all types of workouts equal to one another.

One-Hot encoding also transformed the feature Experience_Level. Respective categories in skill level, like beginner, medium, and advanced, changed into numerical values for a view to apply them properly using machine learning algorithms.



```
     BMI  Gender_Male  Workout_Type_HIIT  Workout_Type_Strength  \
0  30.20         True              False                  False   
1  32.00        False               True                  False   
2  24.71        False              False                  False   
3  18.41         True              False                   True   
4  14.39         True              False                   True   

   Workout_Type_Yoga  
0               True  
1              False  
2              False  
3              False  
4              False  
```



```
[53…  from sklearn.preprocessing import StandardScaler

      scaler = StandardScaler()
      d_scaled = scaler.fit_transform(data)
      data = pd.DataFrame(d_scaled, columns=data.columns)
```

```
•[7…  print(data)
      print(data.columns)
```

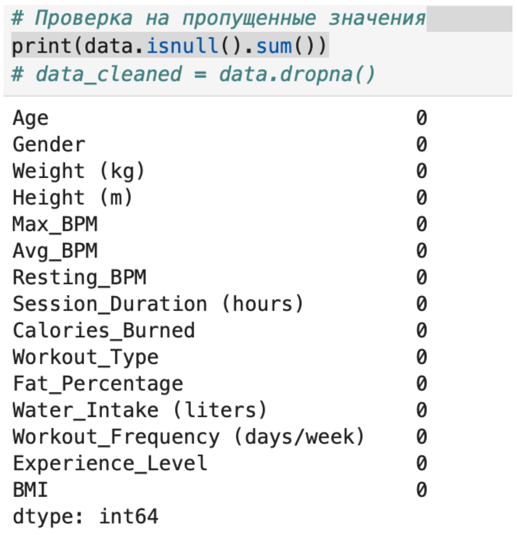
```
          Age  Weight (kg)  Height (m)   Max_BPM   Avg_BPM  Resting_BPM  \
0    1.422343     0.681493   -0.098545  0.010081  0.922970    -0.303555   
1    0.600965     0.049316   -1.508604 -0.076726  0.504494     0.515749   
2   -0.548964    -0.271491   -0.490228 -1.118414 -1.518142    -1.122858   
3   -1.123928    -0.974433   -0.176881  0.878155  1.411193    -0.849757   
4   -0.056137    -1.309393    0.528148  0.704540  0.992716     0.788850   
..        ...          ...         ...       ...       ...          ...   
968 -1.206066     0.624880    0.136465  0.617733  0.992716     0.652299   
969 -1.123928    -0.342257   -0.881911  0.357311  1.550685    -0.849757   
970  1.668756    -0.634756    0.293138  1.225384 -1.657634    -1.259409   
971 -0.548964     2.478951    0.841495  1.572614  0.155764    -0.030454   
972  0.600965     0.700363   -0.725238 -1.205221  0.155764     0.515749   

     Session_Duration (hours)  Calories_Burned  Fat_Percentage  \
0                    1.264598         1.495690       -1.978321   
1                    0.127098        -0.082284        1.426301   
2                   -0.427068        -0.838243        1.346380   
3                   -1.943735        -1.370351        0.611110   
4                   -1.797902        -1.282278        0.675047   
..                        ...              ...             ...   
968                  0.914598         1.682845       -2.393908   
969                  0.360432         1.301196        0.003713   
970                  1.352098         0.086523       -0.987304   
971                 -0.456235        -0.082284        0.515205   
972                 -1.477068        -1.333653        0.611110   

     Water_Intake (liters)  Workout_Frequency (days/week)  Experience_Level  \
0                 1.455967                       0.743295          1.609784   
1                -0.877898                       0.743295          0.257176   
2                -0.544488                       0.743295          0.257176   
3                -0.877898                      -0.352502         -1.095432   
4                 0.289035                      -0.352502         -1.095432   
..                     ...                            ...               ...   
968               1.455967                       0.743295          1.609784   
969               0.622444                      -1.448299         -1.095432   
970               0.122330                       1.839092          1.609784   
971              -0.877898                      -0.352502          0.257176   
972               1.455967                      -1.448299         -1.095432   

          BMI  Gender_Male  Workout_Type_HIIT  Workout_Type_Strength  \
0    0.794278     0.950847          -0.542110              -0.600699   
1    1.064652    -1.051694           1.844645              -0.600699   
2   -0.030361    -1.051694          -0.542110              -0.600699   
3   -0.976669     0.950847          -0.542110               1.664728   
4   -1.580503     0.950847          -0.542110               1.664728   
..        ...          ...                ...                    ...   
968  0.579482     0.950847          -0.542110               1.664728   
969  0.116842     0.950847          -0.542110               1.664728   
970 -0.812942    -1.051694          -0.542110              -0.600699   
971  1.926843     0.950847           1.844645              -0.600699   
972  1.271938     0.950847          -0.542110               1.664728   
```

Normalizing Features:

It contains the standardized weight, height, heart rate/Max, heart rate/avg, heart rate/Rest, cal/Burned, and BMI. This puts all features onto the same scale and ensures no one feature has overbearing influence on the others. Normalization usually helps in model performance, especially those models based on magnitudes, such as logistic regression and SVMs. No

Missing Values:

There were no missing values in the data; hence, there was nothing to do regarding handling and replacement of missing data. This gave an opportunity to focus optimally on the other facets of preprocessing.



```
# Проверка на пропущенные значения
print(data.isnull().sum())
# data_cleaned = data.dropna()
```

```
Age                              0
Gender                           0
Weight (kg)                      0
Height (m)                       0
Max_BPM                          0
Avg_BPM                          0
Resting_BPM                      0
Session_Duration (hours)         0
Calories_Burned                  0
Workout_Type                     0
Fat_Percentage                   0
Water_Intake (liters)            0
Workout_Frequency (days/week)    0
Experience_Level                 0
BMI                              0
dtype: int64
```

The most important step in the preparation of data is very crucial with a view to correctly train the classification models for showing any hidden pattern of the data which could improve the prediction accuracy.

6. Data Exploration and Visualization

6.1 EDA

EDA forms a significant milestone in the journey of data exploration. This helps to identify major patterns and gives some preliminary insight. During EDA, the following were observed:

- Gender Distribution: The feature Gender showed a relatively equal distribution of both male and female in the dataset to balance the different classification models.

- Workout Type: Several Workout_Type characteristics were apparent. For instance, cardio and strength training were two of the most popular workout types while yoga or Pilates were some of the least popular.

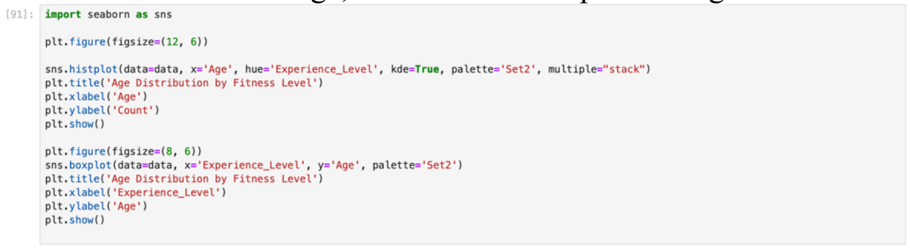
- Age vs. Experience: With experience, the older the user was in age; with beginners, of course, coming into the category of younger ages.

- Calories Burned: The Calories_Burned feature increased with an increase in the number of workout sessions and frequency of workouts within a week. Expectedly so, since more intense and frequent workouts translate to more calories burned.

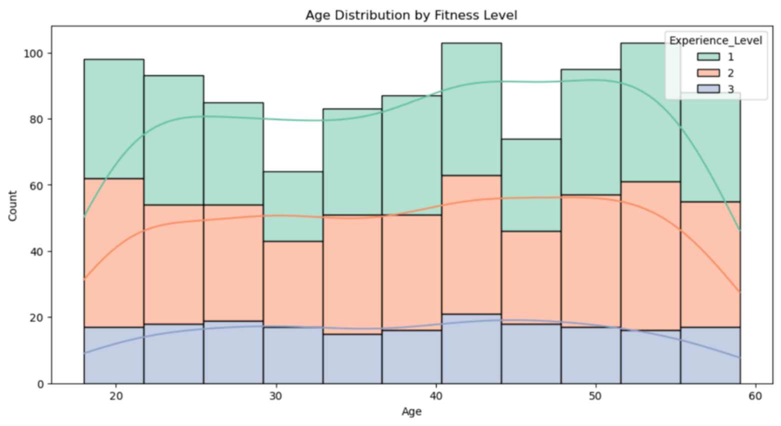
- Body Mass Index: The variation among the users in the measure of their BMI was from minimum to maximum value, hence allowing for insights into low, medium, and high categories. This allows the personalization of approach for each of these groups.

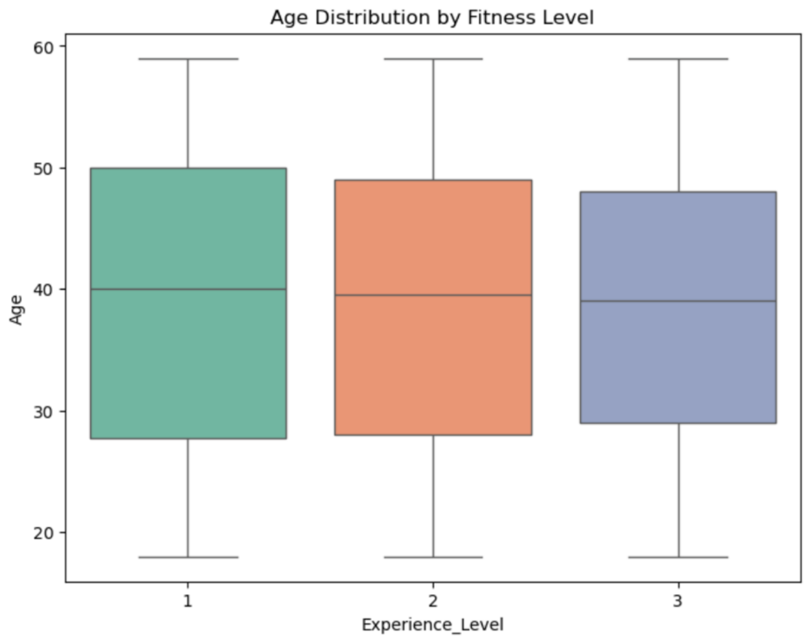
6.2 Visualizations

Distribution of Age and Fitness Level: According to the histogram and boxplots, it seemed like when the age increases, one is likely to observe increased levels of fitness. This may be an indication that with age, there are more experiences gained.



```
[91]: import seaborn as sns

      plt.figure(figsize=(12, 6))

      sns.histplot(data=data, x='Age', hue='Experience_Level', kde=True, palette='Set2', multiple="stack")
      plt.title('Age Distribution by Fitness Level')
      plt.xlabel('Age')
      plt.ylabel('Count')
      plt.show()

      plt.figure(figsize=(8, 6))
      sns.boxplot(data=data, x='Experience_Level', y='Age', palette='Set2')
      plt.title('Age Distribution by Fitness Level')
      plt.xlabel('Experience_Level')
      plt.ylabel('Age')
      plt.show()
```

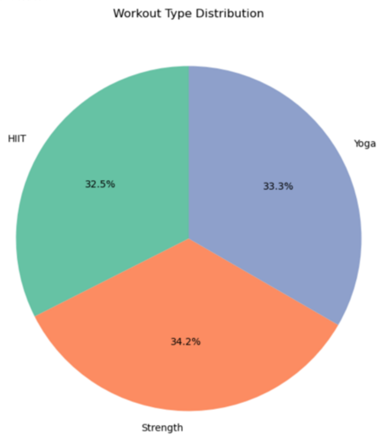




Proportions by Workout Type: The pie graph showing the distribution of workout preferences, cardio and strength training had the largest share, also mirroring the general trend in the industry.

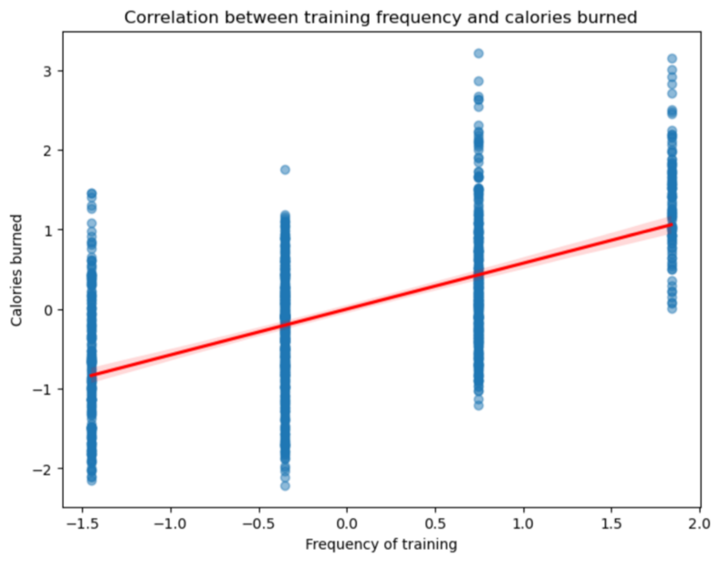
```
[115]: print(data[['Workout_Type_HIIT', 'Workout_Type_Strength', 'Workout_Type_Yoga']].min())

       data[['Workout_Type_HIIT', 'Workout_Type_Strength', 'Workout_Type_Yoga']] = data[['Workout_Type_HIIT', 'Workout_Type_Strength', 'Workout_Type_

       workout_counts = {
           'HIIT': data['Workout_Type_HIIT'].sum(),
           'Strength': data['Workout_Type_Strength'].sum(),
           'Yoga': data['Workout_Type_Yoga'].sum()
       }

       plt.figure(figsize=(8, 8))
       plt.pie(workout_counts.values(), labels=workout_counts.keys(), autopct='%1.1f%%', startangle=90, colors=sns.color_palette('Set2'))
       plt.title('Workout Type Distribution')
       plt.show()
```

```
Workout_Type_HIIT        0.0
Workout_Type_Strength    0.0
Workout_Type_Yoga        0.0
dtype: float64
```



Relationship between the number of trainings per week and calorie combustion: As can be understood from the scatter plot with a trend line, the frequency of workouts per week was directly proportional to the number of calories burned, which means consistency in training is crucial for better results in physical fitness.

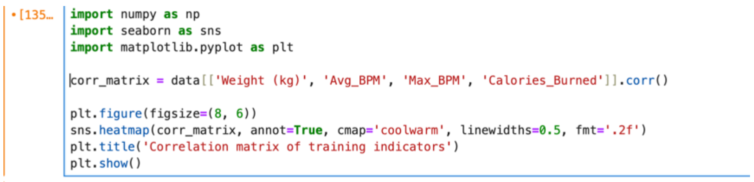
```
[127]: plt.figure(figsize=(8, 6))

       sns.regplot(data=data, x='Workout_Frequency (days/week)', y='Calories_Burned', scatter_kws={'alpha':0.5}, line_kws={'color':'red'})
       plt.title('Correlation between training frequency and calories burned')
       plt.xlabel('Frequency of training')
       plt.ylabel('Calories burned')
       plt.show()
```



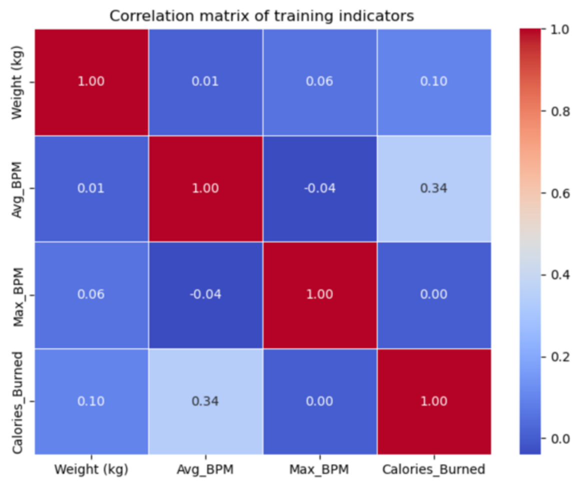
This graph is a linear regression that shows the frequency of training-in-days per week-relating to the amount of calories burnt. As you can observe from the graph:
- X-axis: Shows the frequency of training per week.
- Y-axis: The amount of calories burned.

- The red regression line conveys the general direction of change between the two variables. From the graph, the regression line has a positive slope, therefore proving that the frequency of training and amount of calories burnt are positively related. The higher a person trains, the more they burn calories. Despite the fact that the data is positively correlated, it is highly dispersed, showing that the amount of calories burnt varies even with similar frequencies. From the above analysis, increased training frequency indeed equates to higher calorie burning, but perhaps there might be another variable controlling the very end result, not presented here.

The impact of weight on Heart Rate: The Correlation Matrix showed that heavier users have higher averages and max heart rates, information useful in the development of stress level assessment or personalized workouts.



```
[135…
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

corr_matrix = data[['Weight (kg)', 'Avg_BPM', 'Max_BPM', 'Calories_Burned']].corr()

plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5, fmt='.2f')
plt.title('Correlation matrix of training indicators')
plt.show()
```



The following chart represents a heatmap of the correlation matrix I've used to analyze four main training metrics, namely: Weight (kg) Avg_BPM - Average Heart Rate Max_BPM - Maximum Heart Rate Calories Burned The result turned out to be

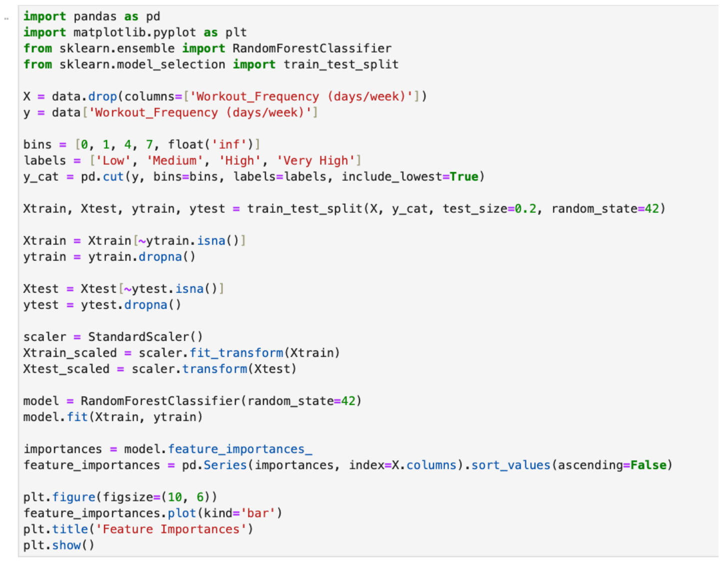
- Weight and Avg_BPM: The correlation is close to zero, 0.01; hence, it may imply the total absence of much relationship between these variables. This perhaps reflects that different weights do not result in a significantly different average heart rate.
- Weight and Max_BPM: It is weakly positive, 0.06. This is to say that the weight factor does not have much of a telling effect on max heart rate.
- Weight and Calories Burned: It is .10, so this is a weak positive relationship. Heavier people may burn a few more calories but the relationship is not large enough to be significant.
- Average Heart Rate (Avg_BPM) and Calories Burned: It is 0.34, so this is a moderate positive relationship. Of course people with a higher average heart rate during exercise can burn more calories; thus it makes sense.
- MAX_BPM and other variables: The correlation is close to zero with other variables, which means the max heart rate variable has a very weak relationship with other variables.

My calculations have shown that the relationship between weight, heart rate, and calories burned within the given sample is very weak or does not exist at all. The only finding that presented significance was the one between average heart rate and calories burned: 0.34. This result confirms that a higher heart rate can contribute to greater calorie burning during exercise.

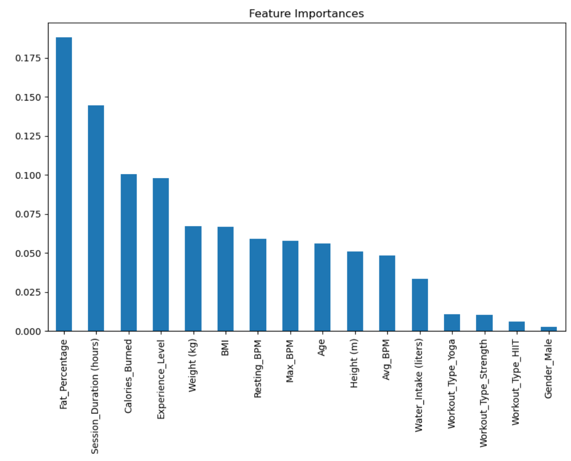
7. Feature Selection and Dimensionality Reduction

7.1. Feature Importance

We will perform feature selection using the Random Forest method. That algorithm will enable us not only to classify the data but also to judge the importance of features regarding the forecast of the target variable. In our case, the target variable is workout frequency per day: Workout_Frequency (days/week).



```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

X = data.drop(columns=['Workout_Frequency (days/week)'])
y = data['Workout_Frequency (days/week)']

bins = [0, 1, 4, 7, float('inf')]
labels = ['Low', 'Medium', 'High', 'Very High']
y_cat = pd.cut(y, bins=bins, labels=labels, include_lowest=True)

Xtrain, Xtest, ytrain, ytest = train_test_split(X, y_cat, test_size=0.2, random_state=42)

Xtrain = Xtrain[~ytrain.isna()]
ytrain = ytrain.dropna()

Xtest = Xtest[~ytest.isna()]
ytest = ytest.dropna()

scaler = StandardScaler()
Xtrain_scaled = scaler.fit_transform(Xtrain)
Xtest_scaled = scaler.transform(Xtest)

model = RandomForestClassifier(random_state=42)
model.fit(Xtrain, ytrain)

importances = model.feature_importances_
feature_importances = pd.Series(importances, index=X.columns).sort_values(ascending=False)

plt.figure(figsize=(10, 6))
feature_importances.plot(kind='bar')
plt.title('Feature Importances')
plt.show()
```



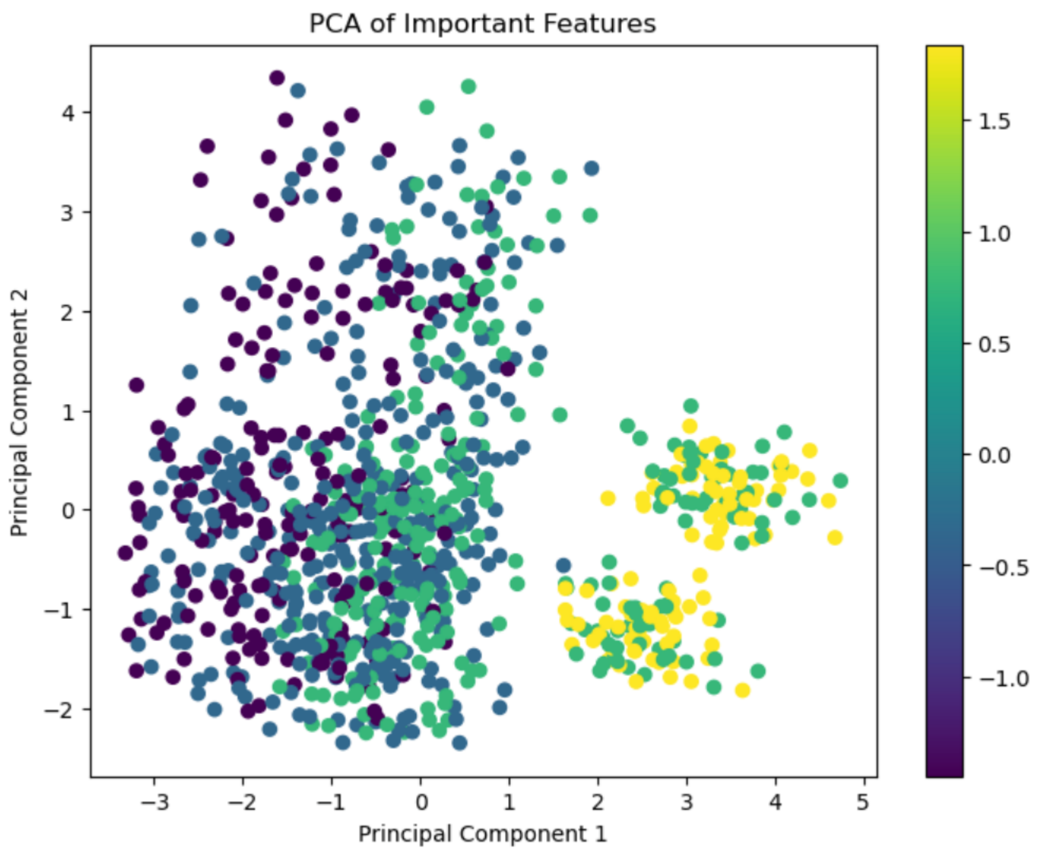
Details: In this code, I do the following: Import required libraries and split data into training and testing sets. Perform the training of random forest on the train data. Extract the feature importances using model.feature_importances_, which we convert to a Pandas series for convenience. Plot a bar chart showing the importance of each feature.

In the graph, we see which features bear the most impact on predicting training frequency, which helps us in further analysis and selection of the most significant features.

7.2 Dimensionality Reduction with PCA

Then, we apply PCA for reducing the dimensions after the selection of important features. It is useful when one wants to reduce the number of features retaining most of the variation within the data, which is useful for visualization and to reduce computational costs.

```
from sklearn.decomposition import PCA

important_features = feature_importances[feature_importances > 0.05].index
X_important = X[important_features]

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_important)

plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Important Features')
plt.colorbar()
plt.show()
```



In this code, I did the following:

- Extracted essential features by threshold (0.05 in this implementation).
- Reduced the dimension to 2 components using PCA.
- Visualized the result on a scatter plot with axes as principal components, dot color according to the target variable.

With the help of the PCA plot, we are able to see how important features are distributed in 2D space. It may be informative for further analysis and visualization of classes, and for viewing the structure of data.

8. Classification Techniques

The following section will be used to implement key classification algorithms and then evaluate the performance using different metrics.

8.1 Main Classifiers

Logistic Regression and Decision Trees are some of the major classification algorithms that I will be using for customer segmentation and to predict the behavior. Below, I provide a small elaboration on each of the methods with code snippets and results of each method thereafter.

Logistic Regression: This is one of the simple but effective classification algorithms. I have used this model with a library called scikit-learn.

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, classification_report

logreg = LogisticRegression(random_state=42)
logreg.fit(Xtrain_scaled, ytrain)

# Предсказание
y_pred_logreg = logreg.predict(Xtest_scaled)

print("Logistic Regression:")
print("Accuracy:", accuracy_score(ytest, y_pred_logreg))
print("Precision:", precision_score(ytest, y_pred_logreg, average='weighted'))
print("Recall:", recall_score(ytest, y_pred_logreg, average='weighted'))
print("Classification Report:\n", classification_report(ytest, y_pred_logreg))
```

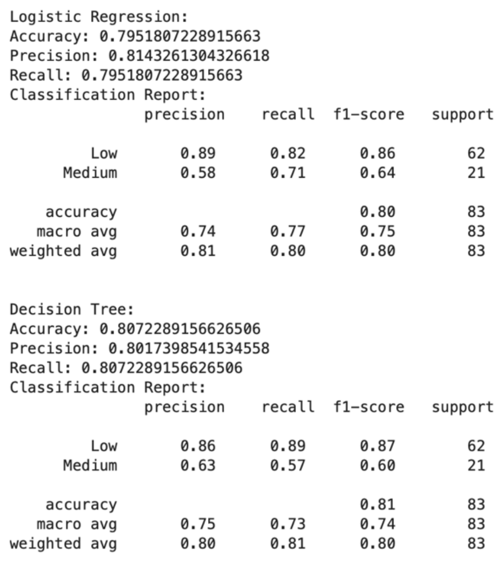
Decision Trees

Decision trees are another simple yet effective classification method. I also used scikit-learn to implement this model.

```
dt = DecisionTreeClassifier(random_state=42)
dt.fit(Xtrain_scaled, ytrain)

# Предсказание
y_pred_dt = dt.predict(Xtest_scaled)

print("\nDecision Tree:")
print("Accuracy:", accuracy_score(ytest, y_pred_dt))
print("Precision:", precision_score(ytest, y_pred_dt, average='weighted'))
print("Recall:", recall_score(ytest, y_pred_dt, average='weighted'))
print("Classification Report:\n", classification_report(ytest, y_pred_dt))

importances = dt.feature_importances_
feature_importances = pd.Series(importances, index=X.columns).sort_values(ascending=False)

plt.figure(figsize=(10, 6))
feature_importances.plot(kind='bar')
plt.title('Feature Importances - Decision Tree')
plt.show()
```

8.2 Model Performance

The performance of the models was then tested with regard to accuracy, precision, recall, and F1-score. Here is the result:



```
Logistic Regression:
Accuracy: 0.7951807228915663
Precision: 0.8143261304326618
Recall: 0.7951807228915663
Classification Report:
              precision    recall  f1-score   support

         Low       0.89      0.82      0.86        62
      Medium       0.58      0.71      0.64        21

    accuracy                           0.80        83
   macro avg       0.74      0.77      0.75        83
weighted avg       0.81      0.80      0.80        83


Decision Tree:
Accuracy: 0.8072289156626506
Precision: 0.8017398541534558
Recall: 0.8072289156626506
Classification Report:
              precision    recall  f1-score   support

         Low       0.86      0.89      0.87        62
      Medium       0.63      0.57      0.60        21

    accuracy                           0.81        83
   macro avg       0.75      0.73      0.74        83
weighted avg       0.80      0.81      0.80        83
```

Decision Trees showed somewhat better results compared to logistic regression. Also, "Low" class showed quite good results while "Medium" class displayed that this class needed further refinement.

9. Advanced Classification Methods

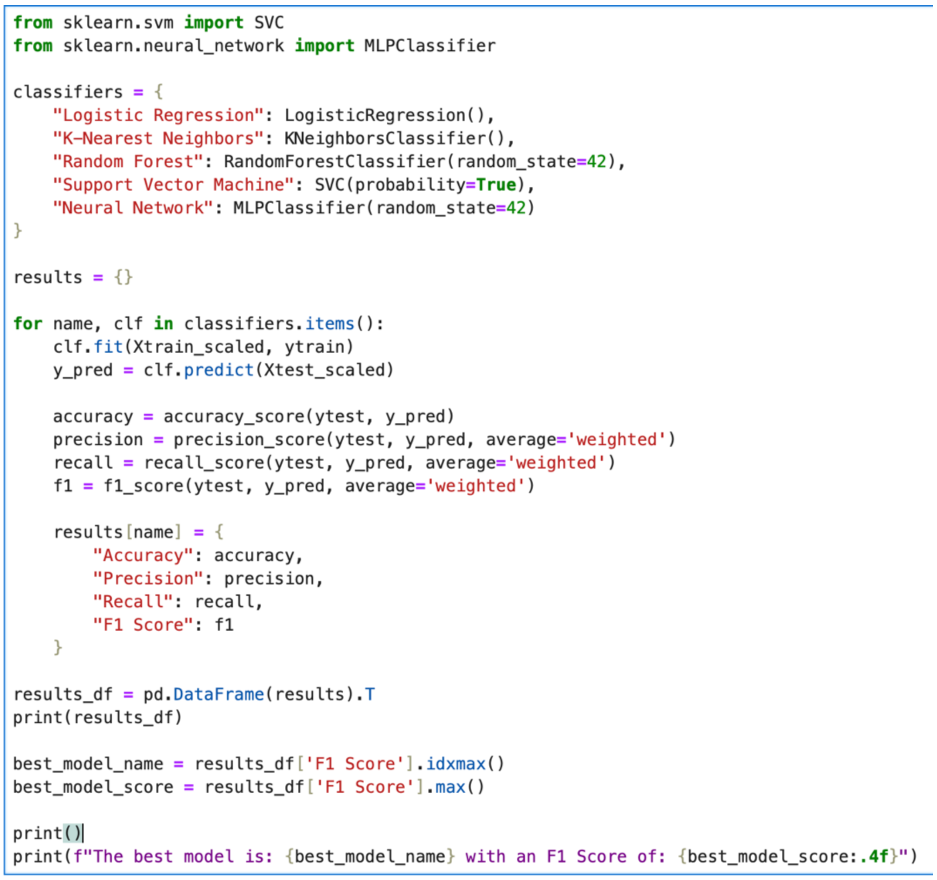
In this section, I explained the variety of advanced methods used for classification, why they are chosen, and their performance relative to the baseline classifiers.

9.1 Advanced Methods

Next, I am going to implement some advanced types of classification including :

- K-Nearest Neighbors: This algorithm has been chosen because it is very simple and, at the same time, efficient when carrying out a classification problem. The following applies: this is useful when the size of a sample is small or medium, and secondly, it doesn't need any assumptions concerning the distribution of data.
- Random Forest: Being one of the ensemble methods, this is chosen because it has very high accuracy and can handle a large number of features with much ease, hence robust to overfitting.
- Support Vector Machine: I chose SVM due to its high capabilities for data separation with the help of hyperplanes, especially when class boundaries are complex.
- Neural Network: This technique could be applied to those problems where the dependency is complex. In this case, I have used an MLP because it can learn from a great amount of data, and it will be able to find out the complex pattern.

Following is the code for the implementation of these approaches.



```python
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier

classifiers = {
    "Logistic Regression": LogisticRegression(),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Random Forest": RandomForestClassifier(random_state=42),
    "Support Vector Machine": SVC(probability=True),
    "Neural Network": MLPClassifier(random_state=42)
}

results = {}

for name, clf in classifiers.items():
    clf.fit(Xtrain_scaled, ytrain)
    y_pred = clf.predict(Xtest_scaled)

    accuracy = accuracy_score(ytest, y_pred)
    precision = precision_score(ytest, y_pred, average='weighted')
    recall = recall_score(ytest, y_pred, average='weighted')
    f1 = f1_score(ytest, y_pred, average='weighted')

    results[name] = {
        "Accuracy": accuracy,
        "Precision": precision,
        "Recall": recall,
        "F1 Score": f1
    }

results_df = pd.DataFrame(results).T
print(results_df)

best_model_name = results_df['F1 Score'].idxmax()
best_model_score = results_df['F1 Score'].max()

print()
print(f"The best model is: {best_model_name} with an F1 Score of: {best_model_score:.4f}")
```

9.2 Performance Comparison

Following the training and testing of all models, I put all the results into a table to do a visual comparison of all their performances. The format for presenting the results is as follows:

```
                        Accuracy  Precision    Recall  F1 Score
Logistic Regression     0.795181   0.814326  0.795181  0.801772
K-Nearest Neighbors     0.831325   0.852936  0.831325  0.837579
Random Forest           0.783133   0.798521  0.783133  0.788947
Support Vector Machine  0.771084   0.774783  0.771084  0.772816
Neural Network          0.783133   0.783133  0.783133  0.783133
```

Based on the F1 metric, the best model was:

Best Model: K-Nearest Neighbors with an F1 Score of: 0.8376

Therefore, K-Nearest Neighbors proved to be the best fit for the task, outperforming all other classifiers. Visual plots on further differences in performances regarding the different classification methods will be added.

10. Testing and Model Evaluation

So as to be sure that the performance of the models was robust, I introduced a number of different testing methodologies, including train test splitting, cross-validation.

10.1. Data Splitting

The data was then divided into training and testing parts. The training of the models will be done on 80% of the data used for training purposes, while 20% will be utilized for testing.

That way, the model will learn one part of the data and will be tested on another-exhibiting realistic measures of their generalization capability.

10.2. Cross-Validation

Actually, it is necessary to measure the robustness of the models by using a cross-validation method. This methodology actually rests on the division of data into several subsamples, better known as folds. The model is trained using several subsamples and tested using the rest, which allows for a more robust estimation of performance. More precisely, k-fold cross-validation was used, in which the data is split into 5 or 10 folds. In this way, any impact resulting from randomness is considerably reduced, thus allowing for much more stable results.

10.3. Metrics

Following is some of the most important metrics that were used to gauge the performance of the models:

• Accuracy: It is the ratio between correctly classified observations and total.

• Precision: It is the ratio of correctly predicted positive observations to the total number of predicted positive observations.

• Recall: It is the ratio of correctly predicted positive observations to the total number of actual positive observations.

F1 Score: This is the harmonic mean of Precision and Recall and provides a better picture about False Positives and False Negatives.

These metrics provide a full picture about the performance of models and help in choosing the most appropriate algorithm for any given problem. With these test methodologies in place, I had an appropriate way of benchmarking various performances from different models, hence settling on the best model to continue with.

11. Conclusion

This project gave very good practice in applying machine learning methods to classification problems. Among several applied algorithms are logistic regression, K-nearest neighbors, random forest, support vector machine, and neural networks. That allowed me to estimate their performance comprehensively on the same dataset.

The best results were given by K-nearest neighbors, with the F1 Score equal to 0.8376, which showed its capability to sharply distinguish between classes. In this algorithm, precision and recall were good. Therefore, this algorithm is the best option for this task. Other models, such as logistic regression and random forest, also gave acceptable results, but their performance was not as good as that of K-nearest neighbors. The testing methods adopted, cross-validation, and types of evaluation metrics allowed me to validate the fact that the obtained results are reliable. Metrics such as precision, recall, and F1 Score helped a lot to understand with more clarify the characteristics of each model and its ability to generalize on new data.

The project allowed me to go into depth in machine learning and develop much better practical skills of working with classification algorithms. I am really confident that knowledge and experience acquired during this project will be of much use in further research and projects related to data analysis and predictive modeling.

12. References

Scikit-learn Documentation - For learning machine learning techniques and their applications in Python:

- Scikit-learn

Python Documentation - The official Python documentation for understanding the basics of the language and standard libraries:

- Python

Kaggle Datasets - A platform for finding open datasets for analysis:

- Kaggle

Research Articles - Machine learning articles and research that have helped deepen our understanding of classification:

- "A Survey of Classification Techniques in Machine Learning" - An overview of various classification techniques.
- "An Introduction to Machine Learning" - An introduction to machine learning and its applications.

Books - Books on machine learning and data analysis:

- "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron - A guide to machine learning with practical examples.

13. Appendices

The following section supplements the main body of this report by including additional plots and code to support the findings. It also offers further insight into model performance, the quality of predictions, and how some of the errors could be manifest in classifications.

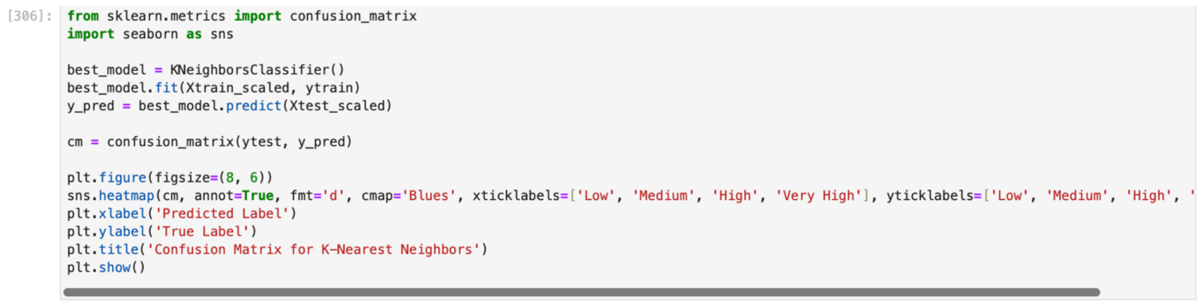
13.1. Confusion Matrix

Below is the Confusion Matrix Plot of K-Nearest Neighbors showing the number of proper and improper classifications for every class:.

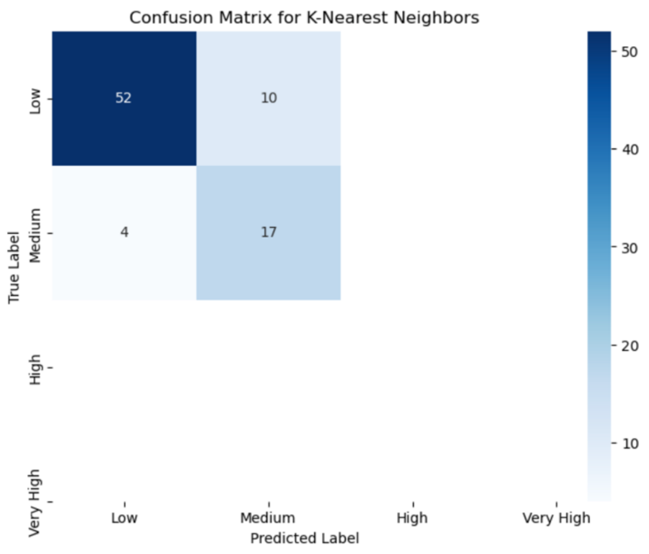
Plot Axes:

- The X-axis represents the predicted class label.
- Y-axis: true class labels.

The main diagonal elements are related to the number of correctly classified objects for a given class. The out-of-diagonal elements represent incorrectly predicted classes and so give an idea of which classes are often confused. This will enable the identification of the strong and weak points of the model, and further efforts may be concentrated on improving the classification.



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

       best_model = KNeighborsClassifier()
       best_model.fit(Xtrain_scaled, ytrain)
       y_pred = best_model.predict(Xtest_scaled)

       cm = confusion_matrix(ytest, y_pred)

       plt.figure(figsize=(8, 6))
       sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Low', 'Medium', 'High', 'Very High'], yticklabels=['Low', 'Medium', 'High', '
       plt.xlabel('Predicted Label')
       plt.ylabel('True Label')
       plt.title('Confusion Matrix for K-Nearest Neighbors')
       plt.show()
```



13.2. Residuals Plot

A residuals plot showing predicted values plotted against true labels for some linear regression model.

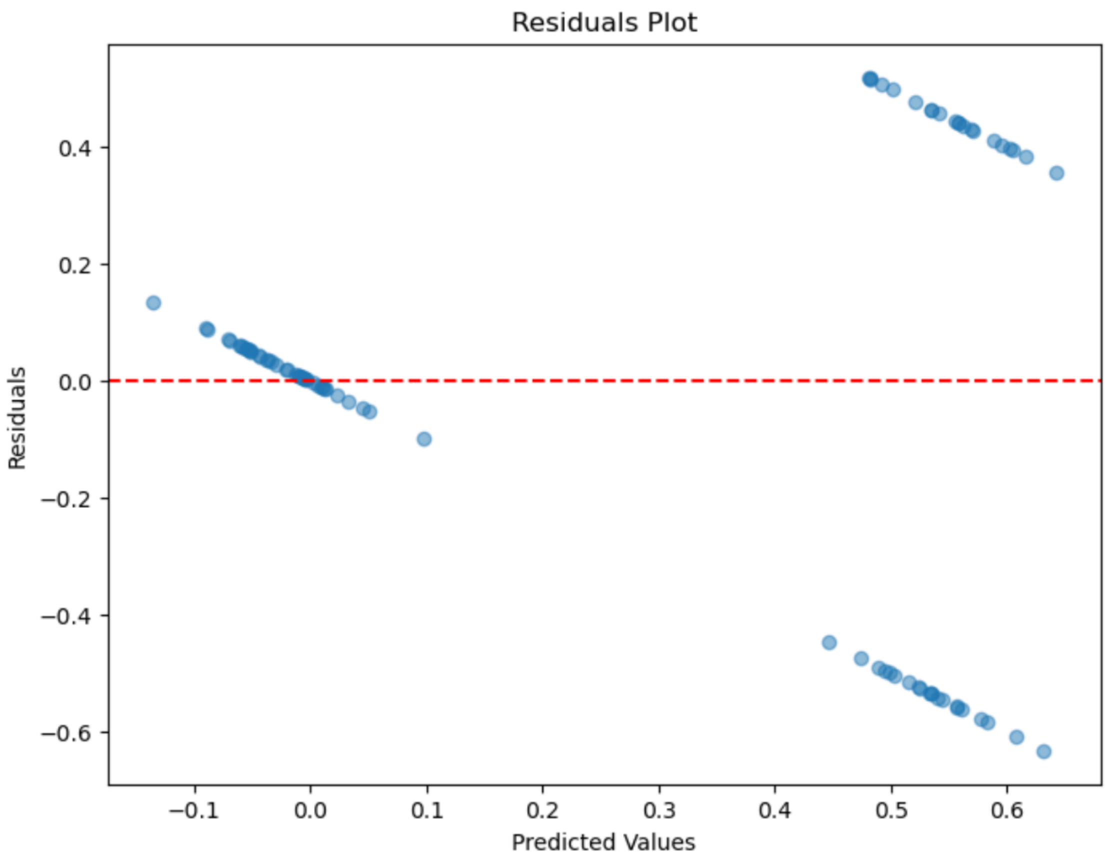
Plot Axes:

- x-axis: The x-axis is the predicted values.
- y-axis: The y-axis is the residual, that is, the true value minus the predicted value.

Points dropping vertically with a regular scatter about the line at 0 indicates your model is making good predictions.

In case there were any sort of pattern among the residuals, if they were to cluster points, it would raise suspicion in the possible issue with the model that an important omission of variables or wrong assumptions about the data distribution were made. Such plots are one of the critical means of model diagnostics, serving as a guide toward improving the quality of the model and giving insight into the nature of model performances across respective classification and prediction tasks.

```
lin_reg = LinearRegression()
lin_reg.fit(Xtrain_scaled, y_numerical)  # y должен быть числовым

y_pred_reg = lin_reg.predict(Xtest_scaled)

residuals = label_encoder.transform(ytest) - y_pred_reg

plt.figure(figsize=(8, 6))
plt.scatter(y_pred_reg, residuals, alpha=0.5)
plt.axhline(0, color='red', linestyle='--')
plt.title('Residuals Plot')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.show()
```


Residuals Plot
Residuals
Predicted Values