Final Exam

PGR210 –3rd semester – 2024

PGR210: Machine Learning and Natural Language Processing

Project Report

Candidate Number 02 & 42

Group Number 2

December 2024

Kristiania University College, Oslo, Norway

Table of Contents

1. Introduction

This project report deals with the basics of Machine Learning (ML) and Natural Language Processing (NLP) which were learned in the course PGR210. Therefore, this report will address three tasks & problems on the Machine Learning (ML) side and two tasks on Natural Language Processing (NLP). All the important theoretical background needed to understand and solve the given tasks are covered in the first section of the paper. This theoretical section does not encompass the entire material and learning outcome that were covered during the autumn 2024 semester, but therefore focuses more on the essential areas necessary to fundamentally understand the tasks & problems addressed in the report, in order to stay within the scope. The three tasks & problems for Machine Learning (ML) are Problem 1: Classification Task, Problem 2: Clustering Task and Problem 3: Deep Learning Based Regression Task are handled in the second section of the paper. The two tasks for Natural Language Processing (NLP) are a Practical Task: Text processing, feature extraction and representation by using both TF and TF-IDF schemes, topic modelling and an Analysis Task: Searching for similar movies. All given tasks for ML and NLP were performed with the given data files which are each described in the task sections. For each task & problem a discussion and conclusion were done at the end of all sections.

2. Theoretical Review

Machine Learning (ML) and Natural Language Processing (NLP) build a good pair of artificial intelligence, enabling systems to derive information from data without the need of programming. ML focuses on learning and recognizing patterns for decision making through learning techniques such as reinforcement, un- and supervised. Whereas NLP equips systems to understand, evaluate, and output human language. Both ML and NLP play a significant role. The combination of ML and NLP helps humans to communicate with computers and to understand the programs in an easier and more efficient way. This helps to drive evolution in the interaction between humans and computers. In order that ML and NLP can work effectively some requirements are needed. Therefore, the most important steps will be shortly explained in the next section. In the whole field of ML and NLP there are of course more techniques and methods than the one mentioned in this report paper. Their use depends mainly on the data sets provided and the problems to be solved. For every ML and NLP processing there needs to be a problem defined, or the data scientists need to define the problem. For example, what is the situation with the defined complication with the goal of finding the solution. For this report the problems were defined in the task (Müller and Guido, 2016).

2.1. Data Preparation

Data in general builds the base of any ML and NLP system. Without data ML and NLP can't work. Therefore, it's important to prepare the data that is most efficient for the processing. Data preparation goes hand in hand with the step of data preprocessing which will be explained in the next step. For this report the needed data was given in excel (csv) files. Otherwise, the data needs to be first gathered and explored by the data scientists. Once the data is fully explored and gathered the step of data preprocessing will come in.

2.2. Data Preprocessing

Data preprocessing is crucial to avoid errors and to maximize the model efficiency & quality of ML and NLP. First, the dataset needs to be cleaned. Therefore, it's important for ML to drop duplicates, handling missing values, numerical features and data imbalance. For NLP it's important to handle the text data, to drop punctuation and stop words. The easiest way to deal with missing values is to drop the samples with missing values. This technique can come with a high risk of data loss, which should be avoided in any case. Therefore, it's recommended to replace the missing values with zero, mean, median or mode. Zero is a base number and in basic datasets it amplifies that the value is absent. If the dataset is not independent of its effect the solution with zero will affect the model. This misleading can be delt with the statistical functions such as mean, median or mode. The calculated numbers represent assumptions and are a solid replacement for missing values (Ghosh, 2023). When dealing with large data sets it's also recommended to split the dataset into two or more sets. One set can be used as a training set for the model and the other one as a test set. This enables a better learning process for the model and avoids overfitting (Buhl, 2024). Next, it's crucial to understand the data in relationships and distribution for further analysis and model building. The best way to describe the distribution is to print a histogram. This shows the relative quantity and range of data values. The calculated presence of outliers, modality, shape, center and modality can help to describe the distribution (*Popov*, 2023).

2.3. Model Building

The process of model building can also be described as a supervised Machine Learning workflow. This building process consists of 6 key steps and teaches the needed knowledge to the model. With that knowledge the supervised model can predict future scenarios. The 1. step is gathering data. Here the supervised learning task is identified and raw data gathered. The 2. step is exploring the data. This step is crucial for understanding the raw data and the distribution

while performing a first analysis and visualization of the data. Next in the 3. step the data is prepared for modelling. This includes dealing with missing values, data wrangling and splitting the dataset into training and test. The fourth step is divided into 4a and 4b. First 4a is for identifying the models and 4b for training the models. In 4a a set of models are collected and trained with the training dataset in 4b. In step 5 the model is tuned. This includes first evaluating the model's performance and afterwards minimizing errors and tuning model parameters. In the last and 6. step the best trained model is chosen and deployed. To identify which model performance is best 4 key model evaluation classification metrics were learned during the Machine Learning course. These were: Accuracy;

$$\mathrm{ACC} = \frac{TP + TN}{TP + TN + FP + FN}$$

The metric accuracy measures the true positives (TP) and true negatives (TN) proportion of correctly classified instances. Recall;

$$\mathrm{Recall} = \frac{TP}{TP + FN}$$

Identifies the ability of the model to choose all important positive instances. Precision;

$$\mathrm{Precision} = \frac{TP}{TP + FP}$$

Calculates the proportion of relevant identified positive instances. F_1score;

$$\mathrm{F_1} = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

The F1score connects recall and precision into a single metric classification and suggesting the balance between these to metrics.

2.4. Analysis

Regarding the analysis of the problems two main methods were learned and applied. First is the regression analysis which explains the relationship between one (linear) or more (multiple) independent and a dependent variable. This is used to predict a certain outcome in the future on the foundation of already measured data. The regression line for example can visualize the mean in the scatterplot. The second one is the hierarchical cluster analysis method. This model works like an algorithm which puts similar objects into clusters. The goal is that each cluster is distinct from the other clusters in the end. Therefore, the clustering can be divisive (top down) and agglomerative (bottom up).

3. Machine Learning

3.1. Problem 1: Classification Task

To address problem 1 in Machine Learning the dataset "Customer Health Profile" with 3500 rows and 5 features were provided. The 5 features are Age, BMI, *Smoking_Habit*, *Exercise_Frequency* and *Blood_Pressure*. The target variable is *Health_Risk*. *Health_Risk* is the depended variable y, while the 5 features (x_1, x_2 … x_3) are the independent variables. These independent variables explain and predict the outcome of the dependent vairable*: Health_Risk*. The dependent variable *Health_Risk* is categorized into three areas: Low, Medium and High Risk. Due to missing values in the dataset the method of K-Nearest Neighbor Method was applied. This method is also called KNN Imputation and deals with the missing values present in the dataset. This method is a machine learning-based imputation which calculates the mean of the nearest neighbors of the missing values for the given row and replaces the missing value with the calculated mean. Therefore the 5 closest datapoints were taken for the calculations. Following this process the numerical data was transformed with the Standard Scaler to get a mean of 0 and standard deviation of 1. In the last step the standardized and normalized dataset was divided into a training and test set for better performance. The balance for this is 70% training and 30% test set. The following 4 figures visualize the confusion matrixes for Support Vector Machine, Random Forest, K-Nearest Neighbors and Logistic Regression.

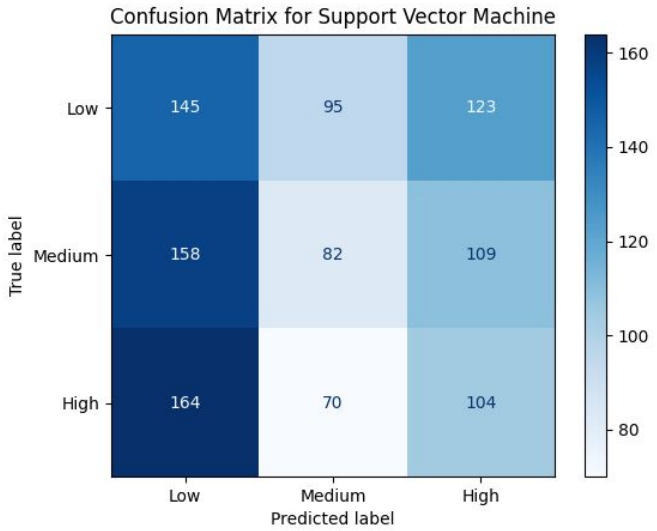

Figure 1: Confusion Matrix for Support Vector Machine

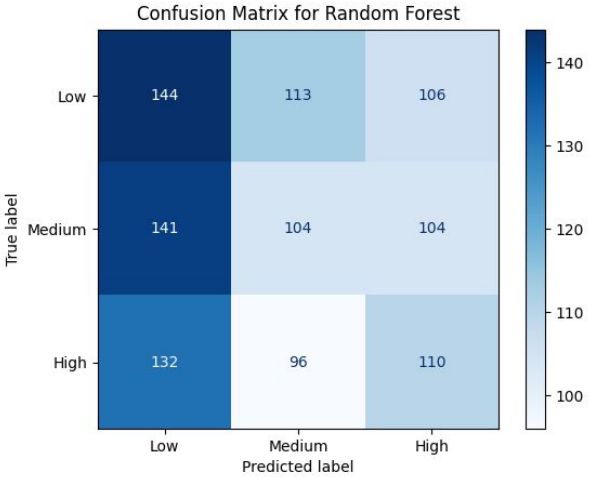

Figure 2: Confusion Matrix for Random Forest

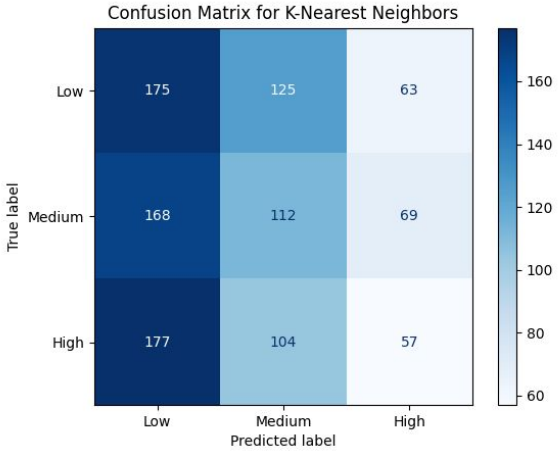

Figure 3: Confusion Matrix for K-Nearest Neighbors

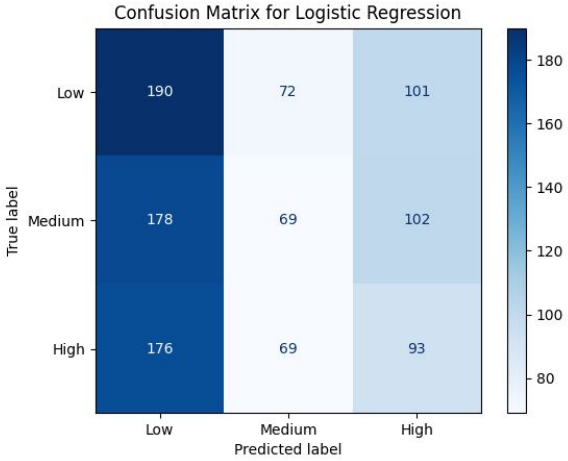

Figure 4: Confusion Matrix for Logistic Regression

Based on the higher accuracy score and more balanced confusion matrix values, we chose to proceed with the Random Forest method for our classification task. For the model building process, the classification algorithm *RandomForestClassifier* was implemented for the given dataset. This algorithm works well on medium-sized datasets like the "Customer Health Profile" dataset. Moreover, this algorithm handles overfitting and missing data very good and works especially well with categorical data, because its an classification algorithm. The model was trained using the earlier defined training set. The algorithm was trained using two training stets *x_train* and *y_train*. To ensure stable results the *random_state* was set to the integer of 42. *X_train* is for the independent values and *y_train* for the dependent value *Health_Risk*. The purpose of the training is to learn the relationship between independent variables and dependent variable. Therefore, the model is trained to make predictions based on the relationship of the variables on the *x_test* dataset. *Y_pred* was used as an output for the predicted variables for the independent target variable y. The built and trained model applies the mathematical system of multiple linear regression.

To visualize the distribution of the *Health_Risk* categories the *Health_Risk* classification model and the different variables were analyzed.

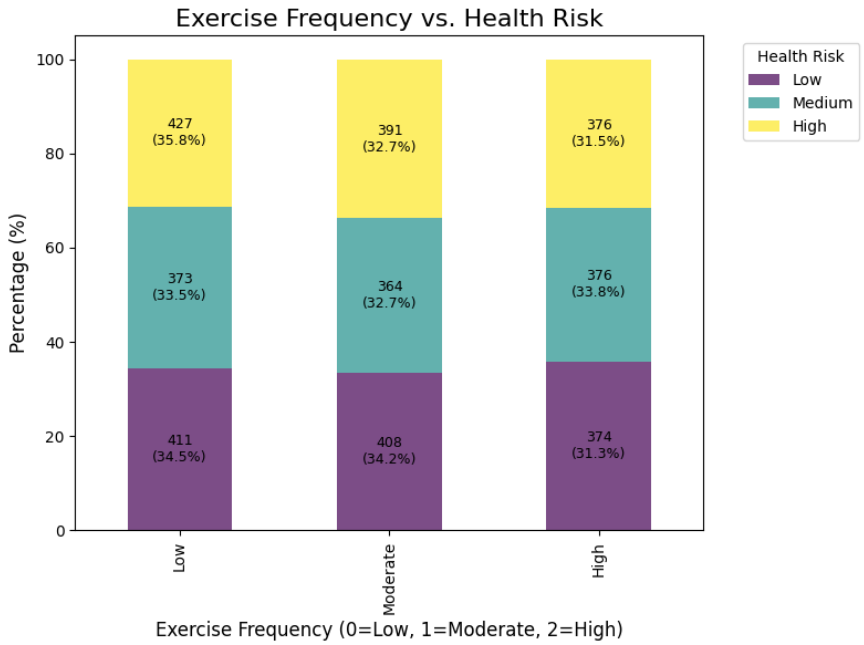

Figure 5: Exercise Frequency vs. Health Risk

Figure 5 shows a stacked bar chart representing the 3 areas of health risk categories (low, medium and high) across different exercise frequencies. Among all exercise frequencies the distribution of health risks is almost similar. This equal level suggests that exercise frequency alone may not significantly describe the dependent variable health risk. The proportions of each

health risk category (Low, Medium, High) across all exercise frequencies (Low, Moderate, High) are similar, which suggests that other independent variables may be more impactful in determining health risk.

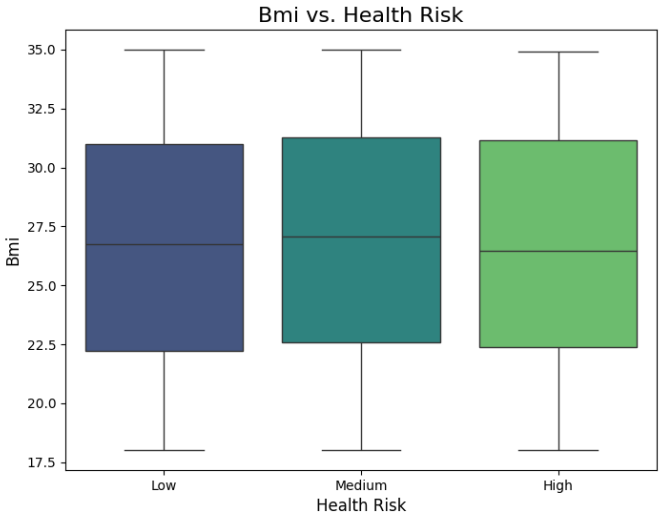

Figure 6: BMI vs. Health Risk

The boxplot in figure 6 shows the BMI distribution across health risks reveals. The BMI median for each health risk category shows an overlap across all categories around 27 to 28. These equal levels indicate that BMI might not be a strong distinguishing factor for health risk in this given dataset. Due to the similar spread of BMI values across the categories, it suggests that BMI may not be the most important predictor for the dependent variable health risk.

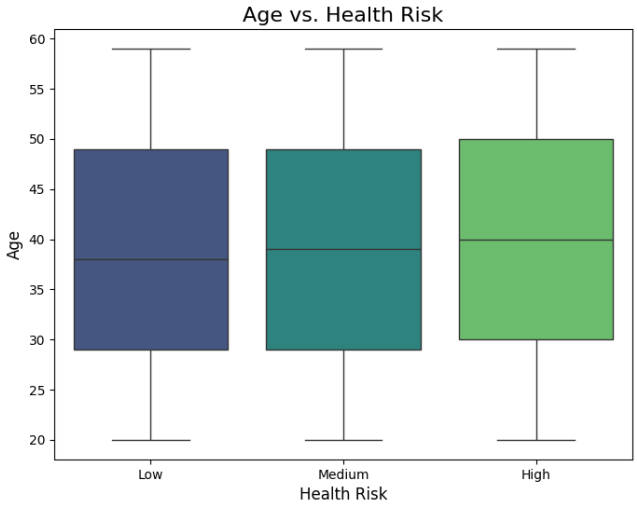

Figure 7: Age vs. Health Risk

Figure 7 shows the boxplot comparing the independent variable age with the dependent variable health risk. Like the BMI boxplot, the chart shows that the distribution of age across all health

risk categories is similar around 38 to 39. Nevertheless, the median age appears slightly higher in the area High, the differences are not large enough though, to indicate a clear trend. This suggests that age may not be the most critical factor in determining health risk in this dataset.

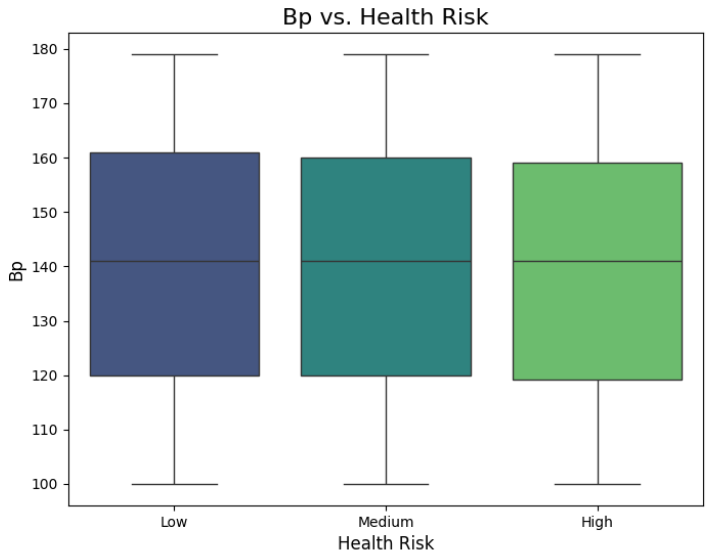

Figure 8: BP vs. Health Risk

The boxplot for Blood Pressure (BP) across health risks shows a similar distribution of BP for each health risk category with a median around 140. Though there may be slight variations, the overall BP levels for the Low, Medium, and High-Risk areas are quite close to the same level. This indicates that BP alone may not be a strong differentiator for health risk in this dataset.

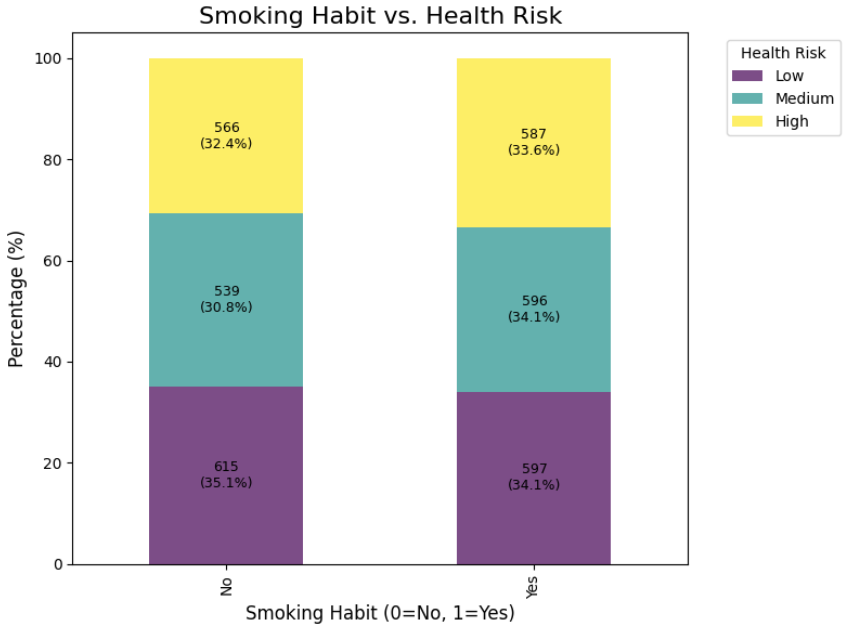

Figure 9: Smoking Habit vs. Health Risk

Figure 9 shows a stacked bar chart for the Smoking Habit across health risks. Among all areas the distribution of health risks is almost similar. In the area with High-Risk a higher percentage of observations compared to nonsmokers can be observed with 33.6% to 32.4%. This suggests that smoking slightly contributes to an increase in health risk but doesn't change the overall distribution of risk areas in this dataset.

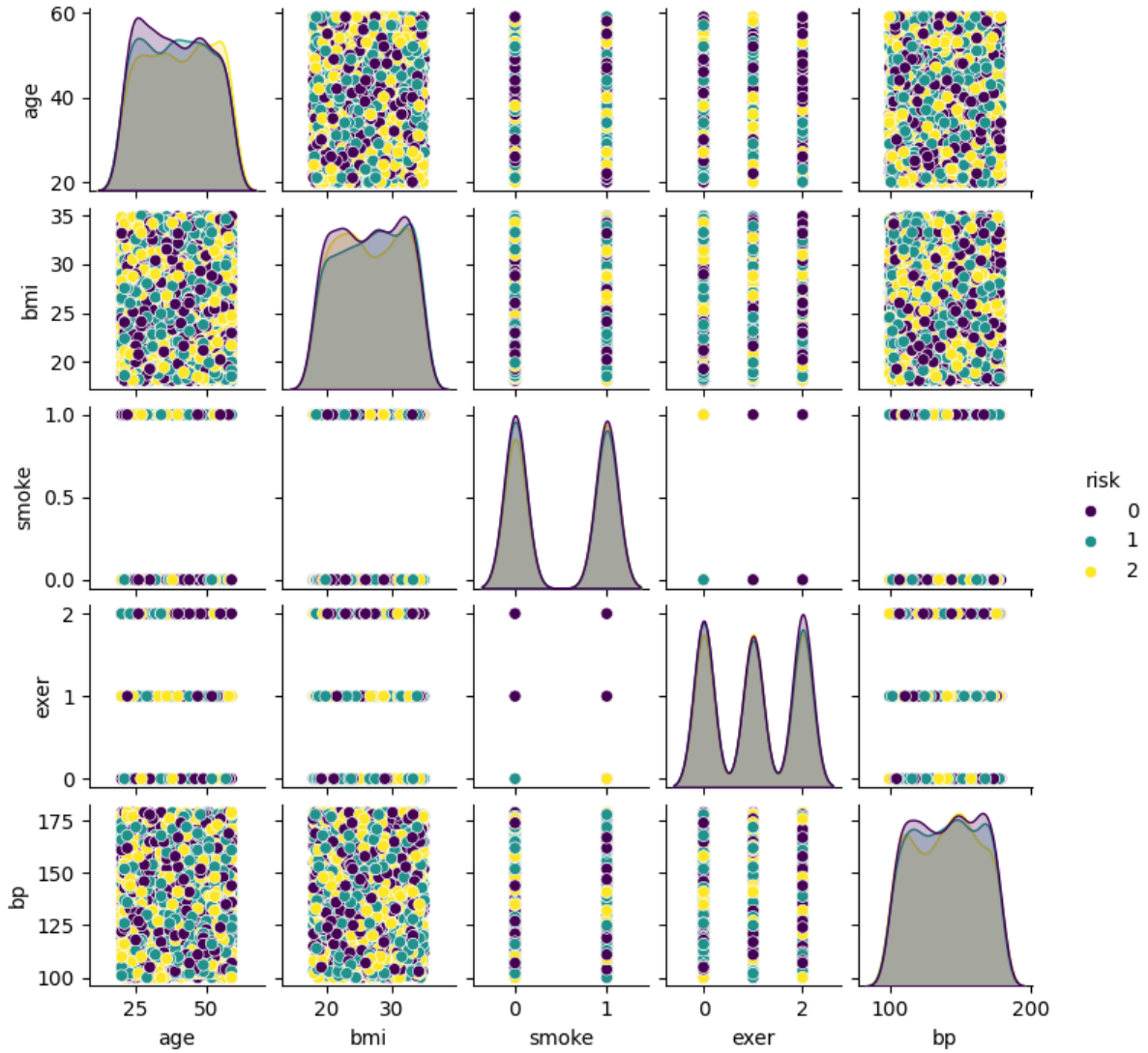

Figure 10: Relationships between Variables

This pair plot shows the relationships between the different variables analyzed earlier. With this visualization the overall distribution is presented. Therefore, the pair plot features to kind of plots, density plots and scatter plots. The risk areas from low to high are here shown with the integer from 0 to 2. For the density plots the variables BMI and BP visualize some differences between the risk areas. Whereas Age, Exer and Smoke show overlapping distributions. The scatter plots show moderate spread across the risk areas for BMI and BP. The other variables Smoke, Exer and Age show slight patterns but do not form strong groupings. This indicates that

not a single variable is a dominant differentiation of health risk, but the combination of all variables is.

Regarding the classification results, the model's accuracy score is 34.29%. This poor score suggests that the model's performance is low and is not clean distinguishing between the health risk areas. Another point is that the dataset can be the problem. The classification report shows the following metrics provided in figure 11.

Metric	Low	Medium	High
Precision	0.36	0.33	0.34
Recall	0.39	0.31	0.33
F1-score	0.37	0.32	0.33

Figure 11: Classification Report

The precision, recall, and F1-scores are all below 40% and around 33% for each health risk area. This suggests that the model is not accurately predicting any of the 3 areas and has a balanced but low performance across all classes. The F1-score being around 33% suggests that the model is failing to capture any meaningful patterns in the dataset. From these classification results and the visualization results it can be inferred that Exercise Frequency, BMI, Age, and BP do not strongly separate the health risk categories in this dataset. The boxplots and bar charts visualize a significant amount of overlap between the categories, suggesting that other factors influence health risk more significantly.

Given the model's low performance, it's recommended to improve the dataset in general. The given data imbalance of health risk areas can be improved with techniques like under sampling, over sampling or the use of class weights to balance the areas.

Overall, some trends can be indicated, but the model performance on the other hand indicates that the current model is not sufficient.

3.2. Problem 2: Clustering Task

The second problem delt with a clustering task. Therefore, the dataset "Online Shopping Behavior" with 6,000 rows and 4 features: *Session_Duration, Page_Views, Purchase_Amount* and *Bounce_Rate* was provided. The goal is to identify different types of shopping behavior. The given dataset has missing values. To handle these missing values, present in the dataset, the method KNN Imputer was used again. First, we renamed some columns to prepare the data easier. Afterwards we filled out the missing data in the set with the method. Originally 5,400

rows were prepared, now with the missing data filled 6,000 cleaned rows were prepared. This method was chosen again, because of its ability to handle mid-sized data sets very good and calculating the meaning of the neighbors without just replacing the missing data with random digits. For the clustering the Elbow Method was used with 5 clusters. The integer 5 was also used for the neighbors in KNN Imputer method. The sum of squared distances within the clusters is plotted against *k* and *k=5* clusters were selected at the "elbow point". Beyond 5 clusters the improvement regarding the cluster compactness was less. With 5 clusters the right balance is achieved. As a suitable clustering algorithm, the K-Means Algorithm was chosen. With the algorithm it was tested where the rate of WCSS (Within-Cluster Sum of Squares) reduction slows, which was around *k=5*. The cluster labels were then added as a new column: *KN-Cluster*.

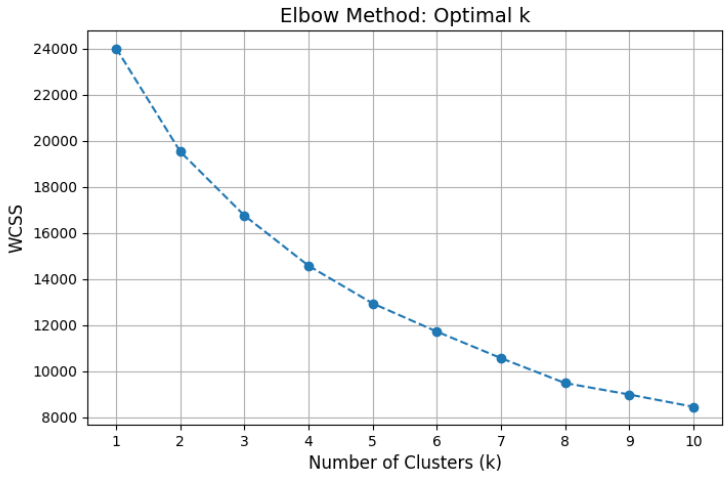

Figure 12: Elbow Method: Optimal k

Figure 12 represents the elbow plot which shows the different numbers of clusters. The elbow as mentioned above is at *k=5*. The number of clusters 5 is in the center of the plot showing that 5 clusters strike a balance between complexity and compactness.

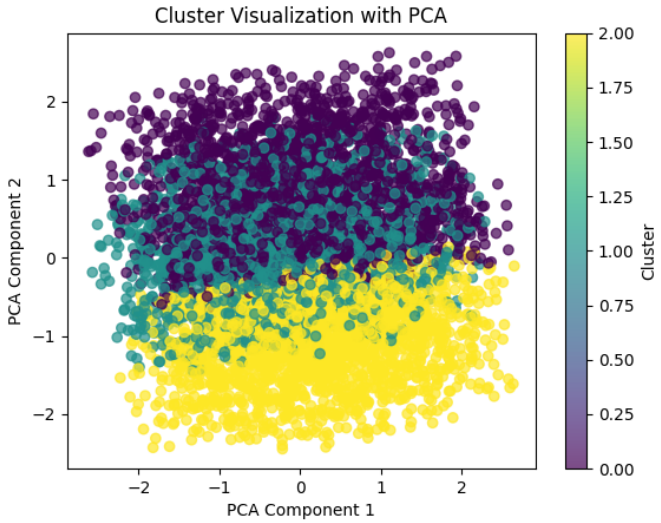

Figure 13: K-Means, k = 3

In figure 13 the K-Means algorithm was run with *k=3*. The data in this chart is grouped into 3 clusters showing a broader distinction between the different groups. These clusters show huge differences in urban mobility behaviors and an unbalanced pattern.

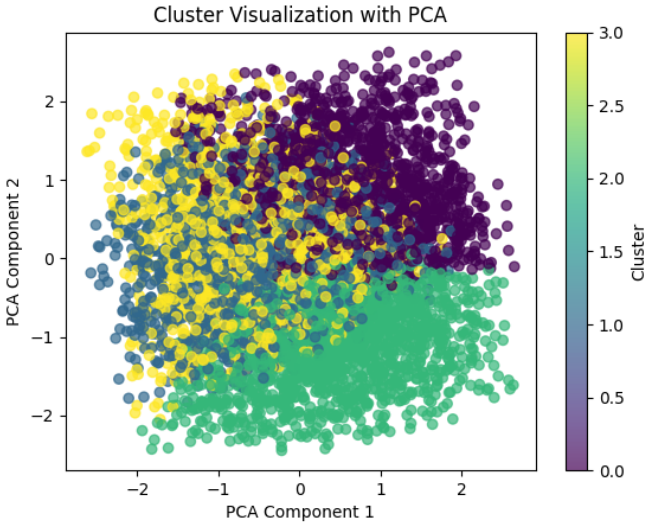

Figure 14: K-Means, k = 4

With *k =4* in the K-Means algorithm the segmentation now becomes more detailed when using 4 clusters instead of 3. The bottom section becomes smaller, and the middle section visualizes cleaner differentiation between the data overall. This indicates an increase in variability regarding urban mobility patterns.

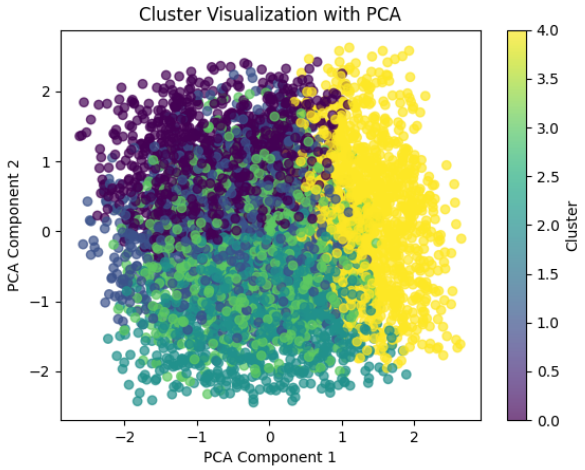

Figure 15: K-Means, k = 5

With one more cluster increasing the total numbers of clusters to 5 more groups or subgroups can be seen. These subgroups represent different behavior regarding urban mobility behaviors. As mentioned before with 5 clusters the optimal number of clusters is achieved as the elbow point. These 5 clusters built the optimal balance of the model.

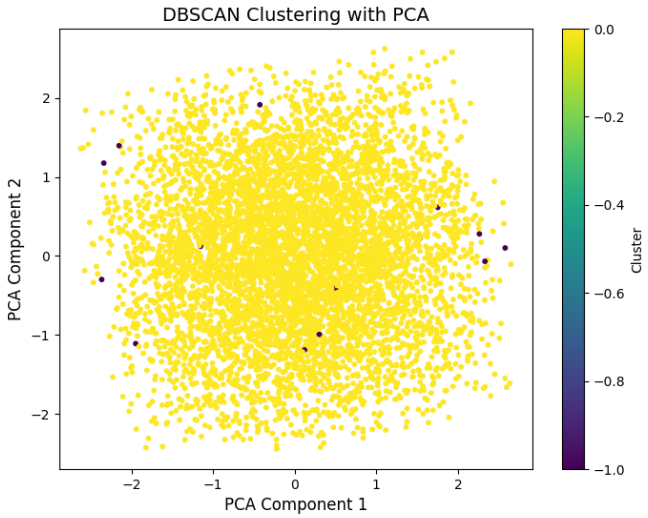

Figure 16: DBSCAN Clustering with PCA

The results of DBSCAN, as shown in Figure 16, revealed one major cluster (label 0) capturing most part of user sessions, while a small amount of data points were classified as noise (label -1). These noisy points represent outliers, such as sessions with unusual shopping behaviors, including extreme session durations, very low page views, or exceptionally high purchase amounts. The results were visualized using PCA (Principal Component Analysis) to reduce the dataset to two dimensions while preserving its structure. The scatter plot in Figure 16 shows that most data points cluster together in a dense region (yellow), with outliers scattered around

the edges, indicating unique or abnormal shopping patterns. While DBSCAN provides valuable insights into edge cases and noise points, its sensitivity to parameter and inefficiency with large datasets limit its scalability. However, it complements K-Means by providing depth to the analysis by revealing edge cases that K-Means might overlook.

3.3. Problem 3: Deep Learning Based Regression Task

The third problem delt with a model training task. Therefore, the dataset “Rental Price Prediction" with 10,000 rows and 6 features: *Property_Area, Bedrooms, Bathrooms, Property_Age, Proximity_to_City_Center* and *Monthly_Rent*. The target variable was Monthly_Rent. The method used to handle any missing values in the dataset was KNN imputation. We chose this method because it is ideal for a mid-sized dataset and we used it before, so it was also implemented in the code twice. After this cleaning step the dataset was scaled using the *StandardScaler* to normalize the variables before training. Before the training part started the dataset was split again into a training set (70%), validation set (15%) and test set (15%) using the command *train_test_split*.

To design a suitable deep learning model for the regression task we built model architecture. The model architecture built is a feed-forward neural network with 4 different layers. The first one is the input layer. The size of the input layer is identical to the number of input features in the dataset. The hidden layer 1 is built of 64 neurons in total. Neurons are basically fundamental building blocks that process and pass information between the different layers of the network. For activation the *ReLU* (Rectified Linear Unit) function was used and helps with problems regarding gradient problems. Hidden layer 2 has 32 neurons and half the size of hidden layer 1. Here the *ReLU* function was used for the same reason as for hidden layer 1. The last layer is the output layer. The output layer only holds 1 neuron, because the model outputs a single value suitable for regression analysis. To measure the average squared difference between real & actual values and predicted value the Loss Function was applied. Also called Mean Squared Error (MSE) to emphasize larger errors in the model architecture. To get different learning rates during testing to optimize the model, the Adam Optimizer was used as an optimizer function.

To achieve the optimal balance between efficiency and performance we trained the testing dataset by doing experimentation with epochs. One epoch presents a complete pass through the entire training dataset to get efficient learning rates. To suggest where the model overfits and underfits we experimented with different epoch amounts from 100 to 1000. With the learning rate the target was to see at which step size the optimizer adjusts the model during training the dataset. The training was stopped when the model achieved the optimal performance on the dataset without overfitting and underfitting.

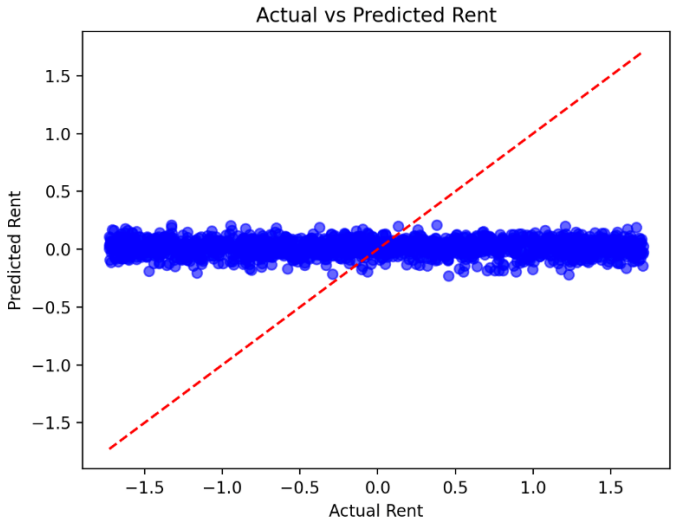

Figure 17: Epoch 100

In the graph the blue points are the predictions, the red line is the regression line. The wide spread of the blue points shows that the model is learning the relationship between the actual rent and predicted rent but fails to show clean patterns in the dataset. This indicates moderate accuracy due to the deviation in the set. We compared the model's performance metrics like Training Loss, Validation Loss, Test Loss, Mean Squared Error (MSE) and Mean Absolute Error (MAE) were compared. Therefore, the Training Loss for the Epoch 100 Result is at 0.9921, Validation Loss at 0.9981, Test Loss at 1.0144, MSE at 1.0144 and MAE at 0.8721. The values of Training Loss, Validation Loss and Test Loss are close, this shows that the model is not overfitting at this point. The high MSE and MAE show that the model has undergone some training but is still not optimal. Next the same test was done with Epoch 500.

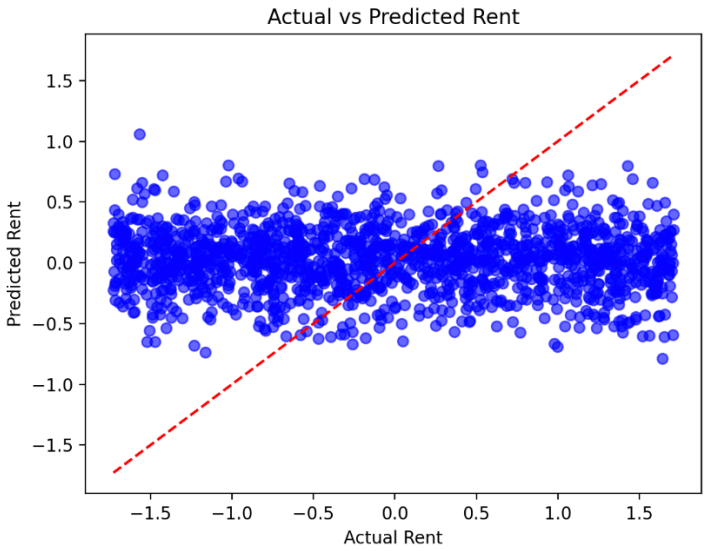

Figure 18: Epoch 500

In the Epoch 500 scatter plot it can clearly be seen that the predictions have a wider spread regarding the predicted rent. Therefore, the predictions align better with the red regression line. Here the model reflects better performance. The same metrics were measured as for Epoch 500. The Training Loss for the Epoch 100 Result is at 0.9108, Validation Loss at 1.0711, Test Loss at 1.0877, MSE at 1.0877 and MAE at 0.8896. Due to the reduced losses and better predictions the model's performance improved compared to epoch 100. The Training loss is below the Validation Loss and Test Loss which suggests signs of overfitting. Last the same tests were done with Epoch 1000.

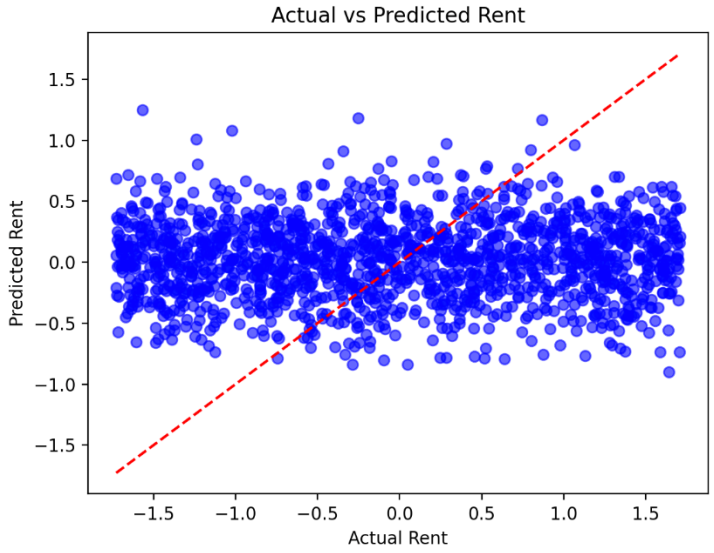

Figure 19: Epoch 1000

In the Epoch 1000 scatter plot the predictions are clustered even wider and spread across the plot. These predictions are more tightly clustered along the red regression line. This suggests a stronger connection between the predicted and actual rent. The reduced outliers show that the model has adapted the patterns more well. The Training Loss for the Epoch 1000 Result is at 0.8712, Validation Loss at 1.0858, Test Loss at 1.1049, MSE at 1.1049 and MAE at 0.8982. The Training Loss has decreased, which means that the model improved the fitting of the data. Due to the increase in Validation Loss and Test Loss the problem of overfitting is now larger than before. The results from Epoch 1000 are clearly more accurate than Epoch 100 and epoch 500 results. Nevertheless, the Epoch 500 underlines a balance between the avoiding of overfitting and model accuracy.

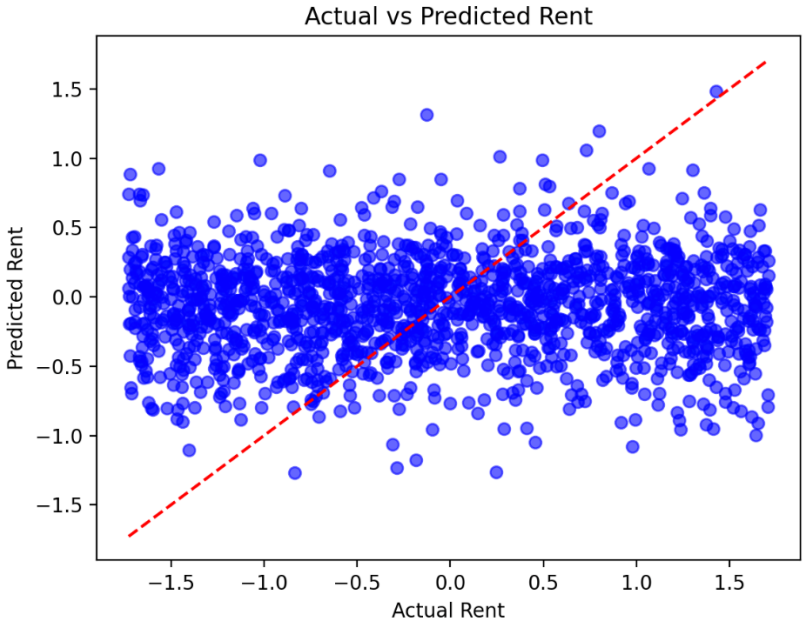

Figure 20: Epoch 500 Learning Rate: 0.01

In the next step the learning rate was tested. Therefore, the constant learning rate was set to 0.01 after experiments with decay schedulers. In total it was experimented with two learning rate schedules: StepLR Scheduler and Final Learning Rate. With the StepLR Scheduler the learning rate was decayed by a factor of 0.1 every 10 epochs. This led to a too small learning rate after some epochs. Then the Final Learning Rate was applied where the learning rate became extremely small during the last Epoch (500th). This rate was *1.0000000000000026e-52* and caused the optimizer to make almost no updates to the model weights, resulting in the loss stagnating. After experimentation, it was settled on using a constant learning rate of 0.01 for the remaining epochs, leading to more stable results. The Training Loss for the Epoch 500 and learning rate at 0.01 Result is at 0.8670, Validation Loss at 1.1245, Test Loss at 1.1495, MSE at 1.1495 and MAE at 0.9089. Figure 16 shows some slight similarities to Figure 15 of the Epoch 1000 Result. The Learning Rate Experiment shows still better alignment with the regression line. The overfitting is at an acceptable level. The use of the constant learning rare of 0.01 provided a stable training process, even though the models performance, especially the test loss, should be further optimized. This could be optimized by testing more Epochs and evaluating more results. Furthermore, more could be experimented with even more learning rates. Regularizations techniques could also be more applied. Especially techniques like early stopping and dropout could decrease overfitting.

4. Natural Language Processing

4.1. Practical Task

The practical task consisted of text processing, feature extraction and representation by using TF and TF-IDF schemes and topic modelling. Therefore, the data file *PGR210_NLP_data1.csv* was provided. The implementation of the topic modelling was covered with Latent Dirichlet Allocation (LDA) and Singular Value Decomposition (SVD). To explore the topic and compare the results from both models the following steps were performed. Step 1. included data loading and preprocessing. The dataset of text messages, consisting of both spam and non-spam messages, was loaded and processed. After this, the text cleaning of the messages. Therefore, special characters, numbers and punctuation were removed. The whole text was then converted to a lowercase. Also stop words to focus more on meaningful words were removed. Next the text was split into individual words (tokens) for further analysis. 2. step was about feature extraction. Here two *CountVectorizer* and *TfidfVectorizer* were used to convert the text into numerical representations. For the *CountVectorizer* a Term Frequency (TF) matrix was created, where each row represents a document, and each column represents the frequency of a term in the model. For the *TfidfVectorizer* on the other hand a Term Frequency-Inverse Document Frequency (TF-IDF) matrix was built, to account for the importance of terms across texts, reducing the influence of commonly occurring words. The 3. Was the topic modelling was where two main methods were used. Latent Dirichlet Allocation (LDA) was used on the TF matrix to identify 5 main topics of the texts. Singular Value Decomposition (SVD) extracted these 5 chosen topics and then were compared with the LDA model. The 5 topics identified by the LDA are represented by their top 10 most repeated words. Topic 1: *ok, lor, time, wat, home, na, going, got, wan, cos*; Topic 2: sorry, da, text, later, hi, pls, send, stop, im, lol; Tobic 3: *free, ur, txt, mobile, reply, claim, prize, new, win, text*; Topic 4*: gt, lt, know, like, don't, want, come, need, day, night* and Topic 5: *ham, love, good, spam, got, day, ur, think, morning, miss*. Regarding the topic distribution, it differed between spam messages and non-spam messages. For spam messages in Topic 1: 459, Topic 2: 79, Topic 3: 56, Topic 4: 26 and in Topic 5: 15 messages were counted. In total, Topic 1: 3592, Topic 2: 251, Topic 3: 180, Topic 4: 176 and in Topic 5: 171 non-spam messages were counted. Topic 1 shows the highest number of spam and non-spam messages with a clean decrease from Topic 2 to Topic 5 regarding both areas. With non-spam messages being higher than the number of spam messages overall. Non-spam messages are more evenly distributed between topic 2 and topic 5. The words which are the top for each of the 5 topics identified by SVD are same as the words identified by LDA. The

different splits of the non-spam messages over the 5 topics are also identical. For the spam messages the amount of spam messages for each topic has changed. In total, Topic 1: 632, Topic 2: 3, Topic 3: 2, Topic 4: 3 and in Topic 5: 2 spam messages were counted. Here Topic 1 is dominant with a huge difference to the other 4 topics. These only contribute minimally which shows poor topic differentiation regarding the spam messages compared to the LDA model. In overall comparison LDA delivers more semantically meaningful topics with more coherent texts. These topics show a better separation of the texts and provide a stronger distinguishing regarding spam messages and non-spam messages. The SVD model shows a poorer topic sparsity than the LDA model with most words clustered in topic 1. Due to the linear decomposition approach the topics are also less interpretable.

In conclusion the SVD model was outperformed by the LDA model regarding this task. The spam message and non-spam message categories were more distinguishable with the LDA model. The LDA model is recommended for the topic modelling regarding this dataset. Further recommendations could be experimenting with additional topic modeling techniques such as fine-tuning the number of topics overall and non-negative matrix factorization (MNF) to improve the separated topics and differentiation.

4.2. Analysis Task

The last task was an analysis task and delt with search for similar movies. First the data was prepared. Therefore, the dataset called: *PGR210_NLP_data2.csv* was loaded into the panda data frame. The data was prepared where missing or null values were checked. Overall, the dataset contains 4804 rows and 20 columns. The most important columns are *genres, keywords* and *production_companies*. The column *description* was created by combining the *tagline* and *overview* columns together. Next the text was preprocessed with the conversion the text to lowercase, removement of any punctuation and special characters, removed stop words and whitespaces. Step 2 was again the implementation of the TF and TF-IDF function. The *CountVectorizer* was used to convert the processed text into a TF matrix. For TF-IDF the *TfidfVectorizer* function was used to build the matrix. In step 3 the cosine similarity was calculated to find similar movies. This was done between the spider-Man movie and all other movies based on their TF and TF-IDF representations.

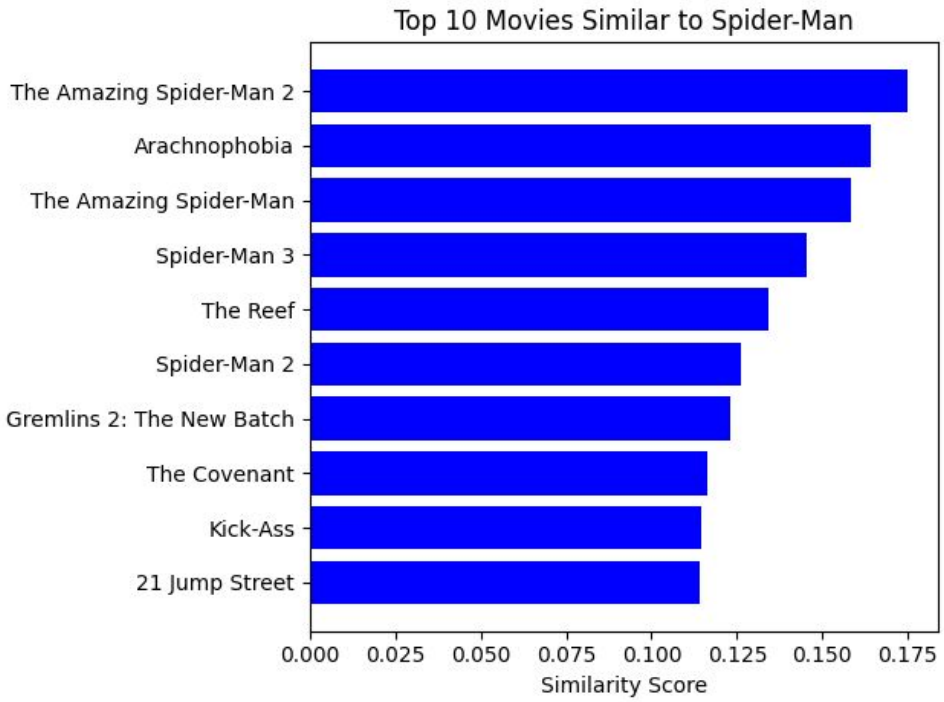

Figure 21: Top 10 Similar Movies

The top 10 similar movies are visualized in Figure 21. With The Amazing Spider-Man 2, Arachnophobia and The Amazing Spider-Man being the closest ones. The results show that Spider-Man itself has a similarity score of 100%. The closest on The Amazing Spider Man 2 has a score of 17,5%. These movies are similar but show lower scores. While the TF and the TF-IDF produce slightly different results, both methods identify The Amazing Spider-Man 2 and The Amazing Spider-Man as the most similar compared to Spider-Man. To find these similar movies to Spider-Man the prepared data and the description column were essential. To compare the different movies the textual description of each movie was converted into numerical vectors. The vectorization was performed using the *CountVectorizer* for TF and *TfidfVectorizer* for TF-IDF. The two output metrics *x_tf* and *x_tfidf* represent the converted values for the movie description. In the next step the similarity between the movies was calculated based on their description. Between the vectors for the Spider-Man and the other movies the cosine similarity was calculated. With this the calculated variable of similar movies were chosen based on the results from 0 to 1. The first ten were then visualized as described in figure 22. The major algorithms used to solve this problem were vectorization algorithms: the TF algorithm and the TF-IDF algorithm. The final and key algorithm used to solve the similarity question was the cosine similarity algorithm.

In conclusion the Amazing Spider Man 2 and The Amazing Spider-Man are the most similar movies. These movies are a sequel of the original Spider-Man, which underlines the result and

makes sense. On the one hand TF focuses on raw word frequency to capture broader similarity of the movies. On the other hand, TF-IDF focuses more on the relative rarity of texts across the dataset. Long story short TF gives more general results whereas TF-IDF provides mor specific results. The most similar movies stay the same for both algorithms.

Overall, the TF and the TF-IDF algorithms successfully suggest movies like Spider-Man, the TF-IDF method is more accurate by modifying the values for less frequent, better informative terms.

5. Literature

Propov, A. (2023) Advanced Methods in Biomedical Signal Processing and Analysis Elsevier eBooks. https://doi.org/10.1016/c2020-0-02228-0. [Accessed: 10th of December 2024]

Advances in parallel computing' (1990) in *Advances in parallel computing*, S. ii. https://doi.org/10.1016/b978-0-444-88621-7.50001-3. [Accessed: 10th of December 2024]

Advances in parallel computing' (1992) in *Advances in parallel computing*, S. ii. https://doi.org/10.1016/b978-0-444-88712-2.50001-9. [Accessed: 09th of December 2024]

Buhl, N. (2024) *Mastering data cleaning & data preprocessing*. https://encord.com/blog/data-cleaning-data-preprocessing/. [Accessed: 11th of December 2024]

Ghosh, S. (2023) *A Comprehensive Guide to Data Preprocessing*. https://neptune.ai/blog/data-preprocessing-guide. [Accessed: 11th of December 2024]

Müller, A.C. und Guido, S. (2016) *Introduction to Machine Learning with Python: A Guide for Data Scientists*. http://cds.cern.ch/record/2229831. [Accessed: 11th of December 2024]

Regression analysis. https://seeing-theory.brown.edu/regression-analysis/index.html#section1. [Accessed: 9th of December 2024]