Data Science - Assignment 1

Jan Hebisch (433925), Franz Link (433731), Ahmad Nour Mammoun (437528)

November 29, 2025

Task 1

(a) We load the dataframe using `pandas.read_csv`. It has 1349 rows and 7 columns.

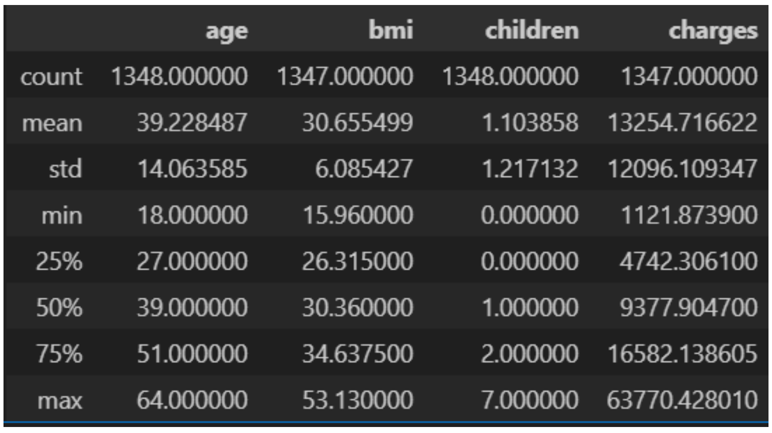
(b) "Explain why there are no basic statistics computed for the other features": Only some features are included because these statistics can only be inferred for numerical features. The screenshot is provided in figure 1.

(c) We remove all rows with `NaN` values using `dropna`, yielding a dataframe with 1338 rows, each without any `NaN` value.

(d) By applying a case distinction to the BMI attribute, we get the following numbers of patients per BMI class. obese: 712, overweight: 392, normal weight: 225, underweight: 20
"Discuss whether you would like or not like to balance out the dataset.": Making up only roughly 1.48%, the underweight category is underrepresented even considering the origin of the data, the United States, but only slightly. If we were to increase that share, we would quickly misrepresent the actual population distribution into the other direction. Nonetheless, assume we absolutely need to avoid poor performance on minority classes. In that case, increasing the proportion of this group becomes vital.
"Name two sampling approaches that were introduced in the lecture that could be used in this case.": In an oversampling approach, we would duplicate samples from the underrepresented group, whereas in an undersampling approach, we would remove samples from an overrepresented group.

(e) We plot the amount of entries per number of children as a seaborn countplot (barplot) in figure 2. The mode is 0 children.

(f) In figure 3, we ultimately choose to use 12 bins, which appears to not be overfit nor underfit to the data, i.e., it maps the most significant aspects of the underlying function's shape without a too complex width for the bins, with each bin accounting for an interval width of *approximately* 5000 dollars. We use multiple = "stack" so the bars do not mix. We determine



	age	bmi	children	charges
count	1348.000000	1347.000000	1348.000000	1347.000000
mean	39.228487	30.655499	1.103858	13254.716622
std	14.063585	6.085427	1.217132	12096.109347
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.315000	0.000000	4742.306100
50%	39.000000	30.360000	1.000000	9377.904700
75%	51.000000	34.637500	2.000000	16582.138605
max	64.000000	53.130000	7.000000	63770.428010

Figure 1: Screenshot of the resulting table showing the basic statistics resulting from the describe() method

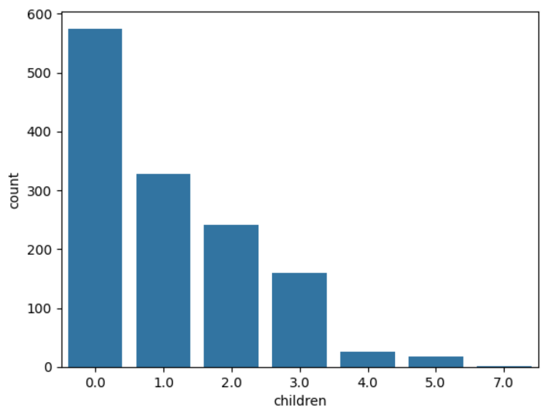


Figure 2: The amount of entries by number of children as a barplot. Mode is 0 children.

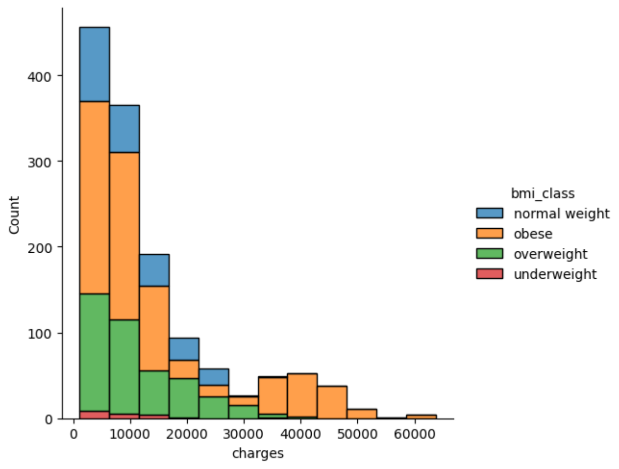


Figure 3: Histogram of the charges feature using binning

that the histogram is multimodal, albeit that the right ”peak” is only slight, so an argument could also be made for ”unimodal, skewed to the left”.

(g) There are trends regarding the charges of patients based on BMI. For instance, obese patients are responsible for nearly all charges over 40000 dollars. Similarly, patients with normal BMI tend to have relatively low charges.

(h) The scatterplot of smokers can be seen in figure 4. ”Report one interesting finding for the relation between charges and the other features considering the smoker feature.”: Interestingly, the plot comparing charges and BMI reveals that, unlike one is left to assume after seeing how obese people are nearly solely responsible for charges over 40000 dollars, obesity alone for non-smokers rarely leads to such costs. It is only the combination of people being obese and smokers that leads to most of these high charges.

(i) Correlation Matrix figure 5 shows: Age most strongly correlates with charges (and vice versa). This can imply that older people tend to have a higher total medical charge, and people with a higher total medical charge tend to be older.

(j) ”Are there more smokers or non-smokers with charges above 20000?”: We cannot be sure. Maybe, the group of non-smokers is really large, and there are more outliers above the upper fence for non smokers (and thus above 20000 in charges) than smokers in the entire dataset. Of course, in an inverse scenario, the box from the smoker plot contains more instances than the entirety of non-smokers.
”Are there outliers for the smoker group that have values below the lower fence value?”: No, because these instances would have to be explicitly marked, yet there are none.
”Is the median of charges for the group of smokers higher than the mean of charges for

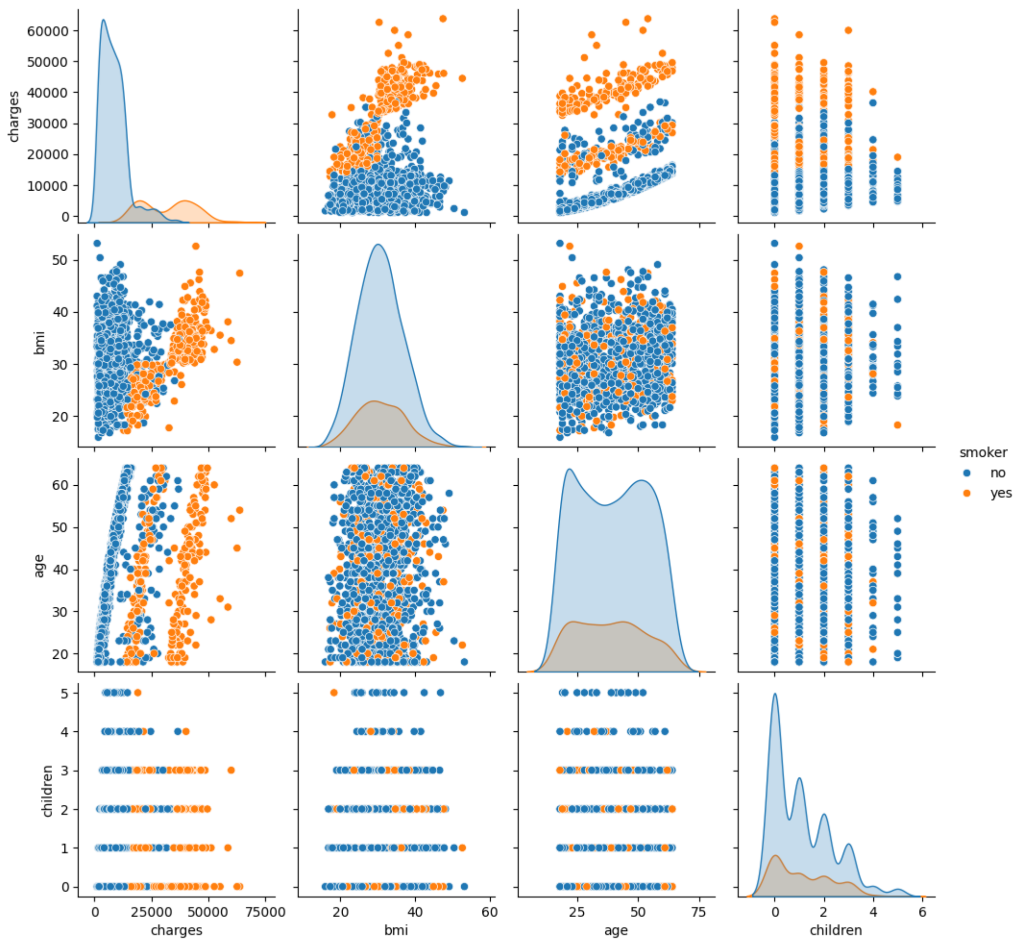
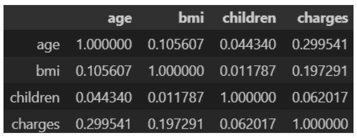


Figure 4: Scatter plot matrix for the numerical features of the dataset.



	age	bmi	children	charges
age	1.000000	0.105607	0.044340	0.299541
bmi	0.105607	1.000000	0.011787	0.197291
children	0.044340	0.011787	1.000000	0.062017
charges	0.299541	0.197291	0.062017	1.000000

Figure 5: Correlations for all the feature pairs from age, bmi, children and charges

the group of smokers?": We cannot answer this, as we do not know how data points are distributed among quartiles throughout the whiskers and the box. If most points were close to the next higher border or whisker, the mean could still be higher than the median, and if most points were close to the next lower border or whisker, the mean could as well be lower than the median.
"Is the lower 1st quartile fence of smokers lower than the upper whisker of the non-smokers?": Yes. The lower first quartile fence is not larger than the lower whisker of smokers. This whisker is clearly below 20000, while the upper whisker of non-smokers is clearly above 20000, thus higher.

Task 2

a)

We constructed a baseline classifier called the $wishfulThinking$ model, which always predicts the class "low", independent of the input features. This model completely ignores the data and serves as a trivial baseline for comparison. To evaluate its performance, we applied it to the test set and computed the accuracy by comparing the constant predictions to the true *chargeGroup* labels. The resulting accuracy reflects the proportion of "low" instances in the test data. Accuracy of $wishfulThinking$ model: 0.49

b)

We implemented a *modeBased* model, which always predicts the most frequent class (*mode*) of the *chargeGroup* feature in the training dataset. After computing the mode, we used it as a constant prediction for every instance in the test set. We then evaluated the model using accuracy. Mode of chargeGroup in the training dataset:

- The most frequent class in the training data is *medium*.
- Accuracy of the $wishfulThinking$ model: 0.49
- Accuracy of the *modeBased* model: 0.39

The *modeBased* model performs worse than the $wishfulThinking$ model. This occurs because the mode of the training data does not match the most common class in the test data. Consequently, always predicting the training-set mode leads to systematically incorrect predictions on the test set. In contrast, the $wishfulThinking$ model always predicts *low*, which aligns better with the class distribution of the test data, resulting in a higher accuracy.

c)

We trained a series of decision tree classifiers with maximum depths ranging from 1 to 9. All trees were configured using the `entropy` criterion, a minimum leaf size of six samples (`min_samples_leaf = 6`), and a fixed random seed (`random_state = 42`). Prior to training, we removed the non-descriptive features `id`, `charges`, and `chargeGroup` to ensure that only relevant predictors were used.

For each tree depth, we evaluated the accuracy on the test dataset. The resulting accuracy curve is shown in Figure 6. The accuracy increases substantially from depth 1 to depth 4, after which it plateaus. The highest accuracy achieved was **0.94** at a maximum depth of **4**. All deeper depths (5 to 9) produce identical accuracies.

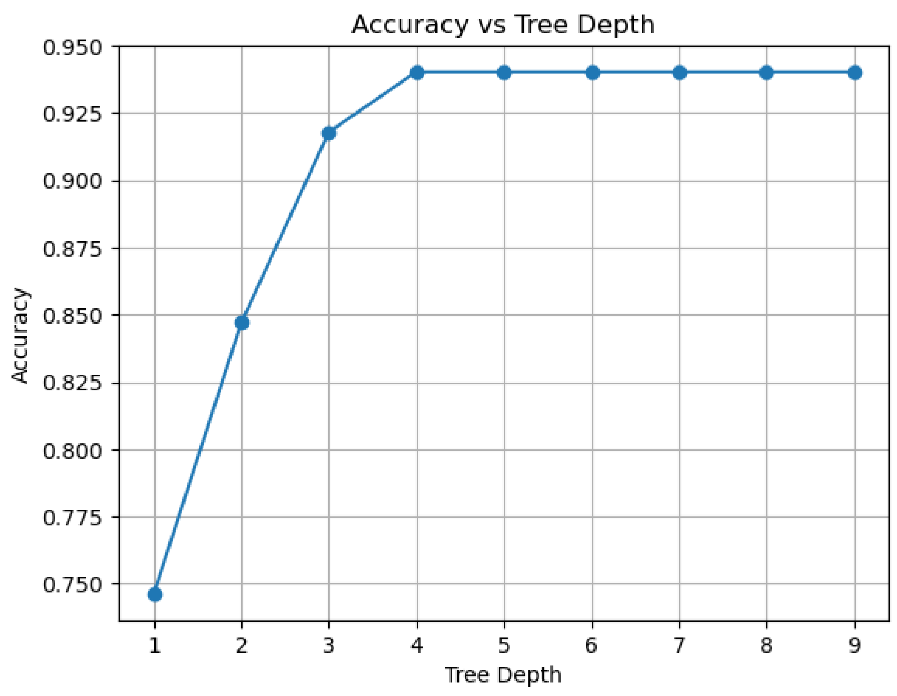


Figure 6: Accuracy of decision trees with varying maximum depth.

This plateau occurs because additional splits beyond depth 4 either violate the constraint `min_samples_leaf = 6` or do not decrease the entropy sufficiently to warrant a split. As a consequence, the effective tree structure does not grow further even when a larger maximum depth is allowed. Based on these observations, we select a maximum depth of 4 as it provides the best predictive performance without introducing unnecessary complexity.

d)

We trained a decision tree classifier to predict the `chargeGroup` feature using the same descriptive variables from the one-hot encoded dataset as before. The model was configured with the Gini impurity (`criterion = gini`), a minimum leaf size of six samples, a maximum depth of two, and a fixed random seed. The resulting tree structure is shown in Figure 7.

From the visualization, we can identify the conditions that lead to a prediction of the `medium` charge group. Since the tree has a maximum depth of two, only two splits determine the classification paths. The leaf node predicting `medium` is reached when the following conditions hold:

- The patient is at most 44.5 years old (`age` ≤ 44.5),
- and the patient is a non-smoker, i.e. `smoker_yes` $= 0$.

Thus, individuals who are younger than approximately 45 years and do not smoke are assigned to the `low` charge group by this tree. No additional conditions are evaluated because the maximum depth restricts the tree to only two decision levels.

We also applied the trained model to a specific example: a 42-year-old male with a BMI of 36, two children, who does not smoke and lives in the northwest region. After constructing the corresponding one-hot encoded feature vector and evaluating it with the decision tree, the predicted charge group was:

$$\texttt{chargeGroup} = \textbf{low}.$$

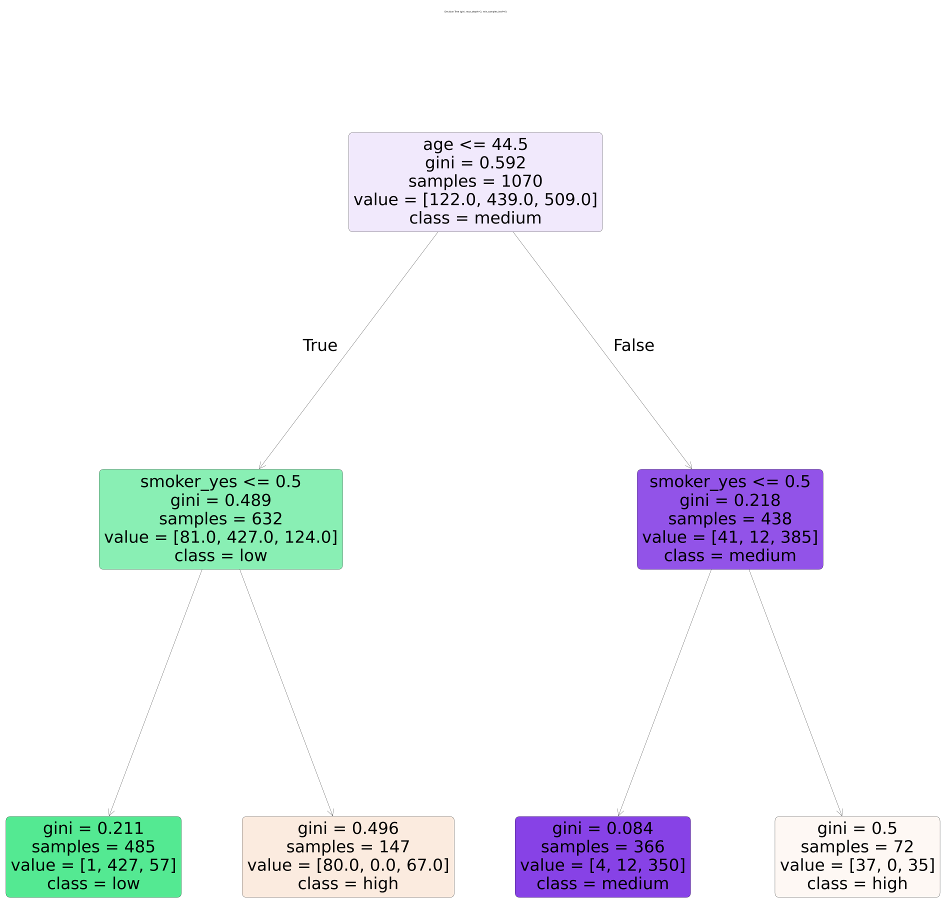


Figure 7: Decision tree for predicting `chargeGroup` with `criterion = gini`, `min_samples_leaf = 6`, and `max_depth = 2`.

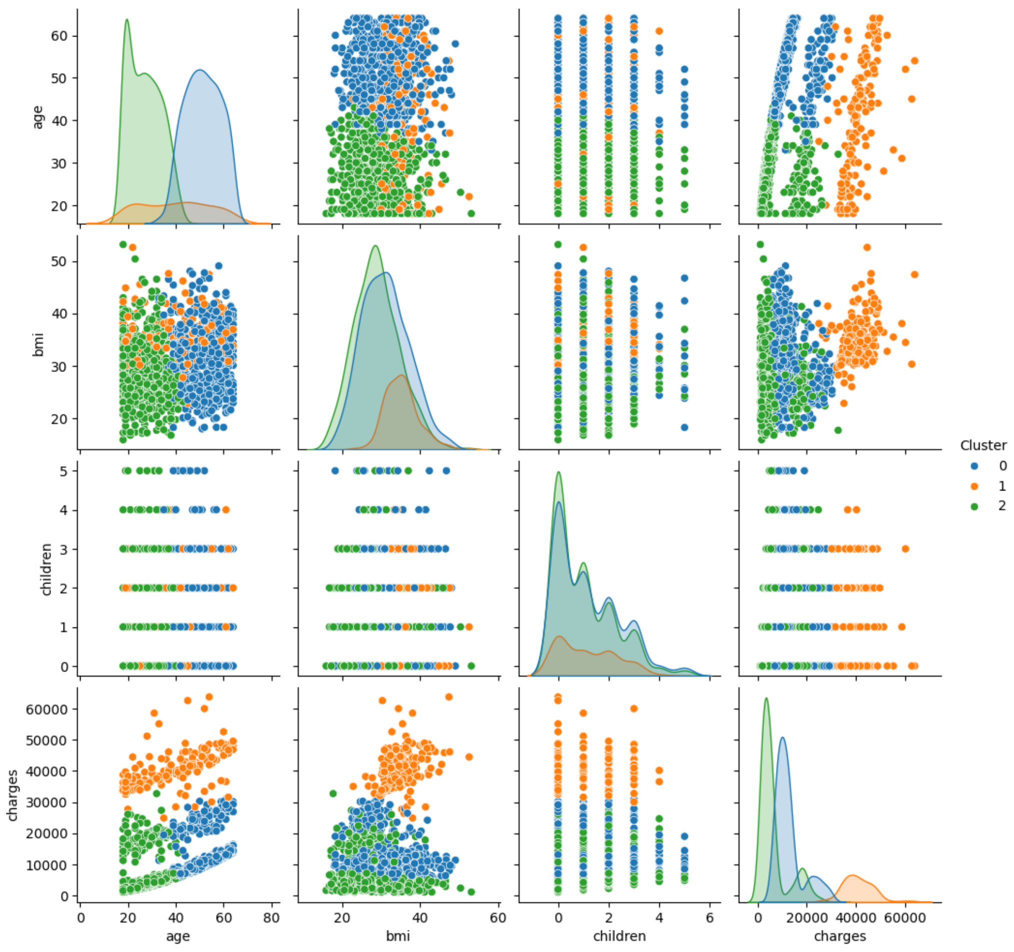


Figure 8: Clusters visualized using a pairplot from the seaborn library

Task 3

The (rounded) centroids are [0.85 0.07 0.06 -0.05], [0.04 0.75 0.04 2.27] and [-0.87 -0.28 -0.07 -0.57].

Cluster 0 is of size 591, cluster 1 is of size 161 and cluster 2 is of size 586.

Cluster 0 has 60 smokers and 531 non-smokers, cluster 1 has 12 smokers and 149 non-smokers and cluster 2 has 65 smokers and 521 non-smokers. In 8, we observe that cluster 1 is predominantly seperable as the higher-cost group, as the bottom right plot shows. Cluster 0 and 2 most notably differ in age, as seen in the top left. Correspondingly, points from cluster 1 are mostly distinguishable from the rest when relating an attribute to charges, whereas elements from cluster 0 and 2 are most distinguishable by age, as indicated by the top left plot, showing that age and charges made the most vital impact on the positions of the centroids.

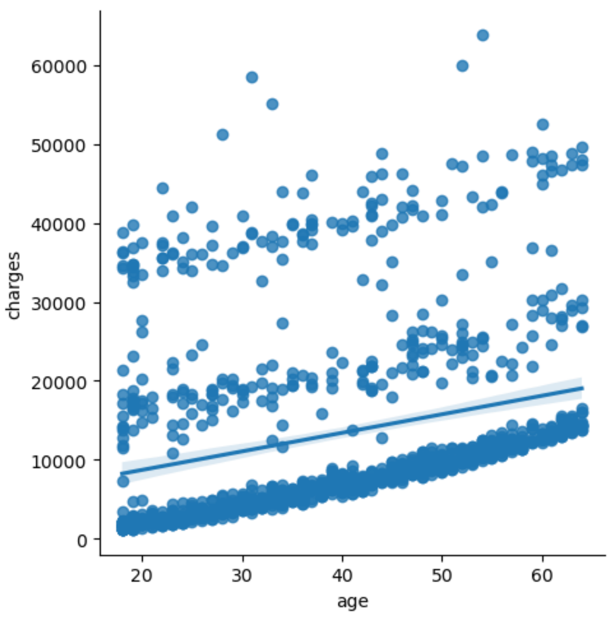
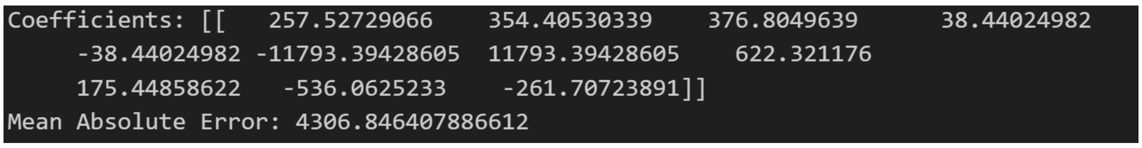


Figure 9: Linear regression of `charges` against `age` on the training data.

Task 4

(a) Charges over age. We visualized the relationship between `age` and `charges` on the training set using `seaborn.lmplot`, including the fitted linear regression line (see Figure 9). The plot and the linear regression line both show a slight positive trend: on average, higher age is associated with higher charges. However, the points are widely scattered and form several bands, indicating that age alone cannot explain most of the variance in medical costs. Hence, a simple linear regression on age only is not a particularly good fit for this data.

(b) Linear regression model for charges. Next, we trained a multiple linear regression model using the `LinearRegression` class from scikit-learn. As input features we used all columns of `train_insurance_one_hot.csv` except `id` and `chargeGroup`; the target variable was `charges`. After fitting the model, we obtained the following coefficient as seen in the firgure 10



```
Coefficients: [[   257.52729066    354.40530339    376.8049639      38.44024982
    -38.44024982 -11793.39428605  11793.39428605    622.321176
    175.44858622   -536.0625233    -261.70723891]]
Mean Absolute Error: 4306.846407886612
```

Figure 10: Coefficients and MAE of the linear regression model.

and evaluated the model using the test set. The mean absolute error (MAE) on the test data was **4306.85**. This error indicates that, on average, the predicted charges deviate from the true charges by about $4300, which reflects the substantial variability and complexity of the underlying cost structure that cannot be fully captured by a linear model.

	expensive	not expensive
expensive	36	9
not expensive	3	220

(a) SVM (1)

	expensive	not expensive
expensive	36	9
not expensive	3	220

(b) SVM (2)

Table 1: Confusion matrices for SVM (1) and SVM (2).

Task 5

Before the start of any subtask, the data is prepared. To that end, the one-hot-encoded training and test data sets are loaded. Then the new target feature `expensive` is created according to the task description. Note, that these steps where already completed in the provided Jupyter file. Afterwards, the columns of features `charges` and `chargeGroup` are removed from the training and test data set. Finally, the target vectors for training and test data are created.

Subtask a)

For SVM (1), we use the descriptive features `age` and `bmi` as described in the task and the descriptive feature `smoker` in the form of both `smoker_yes` and `smoker_no` that are features in the one-hot-encoded data sets. We decided to use both of the one-hot encoded features of the original feature `smoker` because it fits the task description the most and does not have a noticeable impact on model performance. For SVM (2), we use all descriptive features expect for the once excluded in the task description. Regarding the origin features that are encoded in multiple features, we decided to use all of them for the same reasons as before.

For both SVM (1) and (2), we created a new SVM each using `SVC` of `sklearn.svm` with regularization parameter $C = 10$ and `kernel="linear"` since the task calls for linear SVM classifiers. The SVMs are then trained using the training data constrained to the respective descriptive features as described before. For training, we use the method `fit()`.

Now that SVMs (1) and (2) are trained, predictions of the target features are created based on test data constrained to the descriptive features of SVM (1) and (2) respectively. This is done using the `predict()` method.

Using `confusion_matrix` of `sklearn.metrics`, we compute the confusion matrices for the prediction of SVM (1) and (2). Find these in table 1. A detailed explanation follows in subtask (b).

Subtask b)

The confusion matrices in table 1 created for subtask (a) will now be discussed. The first thing we notice it, that both matrices are exactly equal despite differences in the descriptive features used to train SVM (1) and SVM (2). Because SVM (1) achieves the same prediction quality as SVM (2), we assume that the descriptive features `age`, `bmi`, and `smoker` (in the form of the one-hot-encoding discussed in subtask (a)) are as a whole or in parts sufficient to predict whether a charge is `expensive`. This assumption is supported when looking at the descriptive feature `smoker` and the corresponding distribution of target features. `smoker` alone already allows for a prediction close to the predictions of SVM (1) and (2). Note, that we only looked at the raw data for this result and counted using the basic document search function and did not train further SVMs on different descriptive features.

The other descriptive features `age` and `bmi` do not allow such an easy observation by eye. This provides an explanation for the behaviour of the SVMs regardless and thus fits our expectations even though an intuitive expectation might assume better behaviour of SVM (2) as it uses more descriptive features.

The accuracy and precision of both SVM (1) and (2) are equal as the confusion matrices are equal. The accuracy of both models is $\frac{36+220}{36+220+3+9} \approx 0.96$. This indicates that the over-all quality of our SVMs is high.

The precision of both SVMs is $\frac{36}{36+3} \approx 0.92$. This precision still indicates a good classification quality, but also shows where our models might perform slightly worse. The models generally

	expensive	not expensive
expensive	36	9
not expensive	3	220

(a) Descriptive features as in SVM (1)

	expensive	not expensive
expensive	36	9
not expensive	2	221

(b) Descriptive features as in SVM (2)

Table 2: Confusion matrices for SVMs with normalized data and the same descriptive features as SVM (1) and SVM (2).

make comparatively more mistakes classifying as `expensive` compared to the overall correctness of classification.

Subtask c)

We use the `StandardScaler()` of `sklearn.preprocessing` with default parameters for normalization. To this end, the entire training data is first fitted to the scaler using `fit()`. Then, we normalize the entire training and test data using `transform()`. We also recreate the previous structure of the data by adding the column names anew using `pandas.DataFrame()`. Note, that we do not normalize the target feature as our SVMs are binary classifiers.

Now, we follow the same steps as in subtask (a) to train and test SVMs on the descriptive features of SVMs (1) and (2) using the normalized data sets. In this way, we get the confusion matrices in table 2.

When comparing the SVMs using descriptive features of SVM (1) with and without normalized data sets, we see that the confusion matrices are equal once more. The reason for this is likely that the descriptive features `bmi` and `age` have a somewhat similar value range and that smoker is binary.

The SVM with the descriptive features of SVM (2) and normalization behaves minutely better than SVM (2). A single false positive less occurs with normalization increasing the amount of true negatives by one. Otherwise, the confusion matrices of these SVMs are also the same. This behaviour has likely similar reasons as previously. The only feature that has a slightly different range and is not binary is `children`. Because the range of `children` is not drastically different from the other parameters, normalization likely only leads to the minute improvement.

The training and testing for SVMs (1) and (2) takes $0.1s$ and $0.2s$ respectively. Using scaling prior to training with the same descriptive features reduces this time such that the jupyter notebook indicates $0.0s$ as computation time. This decrease likely results from a fitting process with scaled data that can be performed faster.

Subtask d)

To investigate the impact of decreasing the regularization parameter C, we performed training and testing as described in subtask (a) for the values of C in the first column of 3. We used `accuracy_score()` of `sklearn.metrics` to compute the accuracy of both SVM (1) and (2) with the different regularization parameters as depicted in the second and third column of the same table.

No changes of accuracy (with the given precision) occur until $C = 0.50$ for SVM (2). Then, we notice a local minimum between $C = 0.50$ and $C = 0.10$. This minimum is not reflected in SVM (1). Therefore, the decreased accuracy is likely caused by the increased amount of descriptive features. Based on the data, we assume that the descriptive feature relevant for correct classification of the test data are disregarded to a higher degree which is permissible due to the lower value and by chance optimal.

C	SVM (1)	SVM (2)
10.00	0.96	0.96
7.50	0.96	0.96
5.00	0.96	0.96
2.50	0.96	0.96
1.00	0.96	0.96
0.75	0.96	0.96
0.50	0.96	0.95
0.25	0.95	0.94
0.10	0.96	0.95
0.05	0.95	0.96
0.03	0.94	0.96
0.01	0.86	0.87

Table 3: Accuracy of SVM (1) and (2) for decreasing regularization parameters C.

	`low`	`medium`	`high`
Mean	0.41	0.47	0.12
Variance	0.00	0.00	0.00

Table 5: Mean and variance of probability estimates for the MLP with 5 hidden layers of 5 neurons each using logistic activation function.

For SVM (1), the accuracy of the model begins to decrease starting at $C = 0.05$. Before a similar trend can be noticed for SVM (2), the accuracy first goes up to the value of $C = 10$ for $C \in \{0.03, 0.05\}$. We attribute this behaviour to the specific instances in test and training data.

This decrease in performance for low enough regularization parameters is expected. The trade-off between classification quality and punishment for errors with small values of C at some point disregards the error to a high enough decree that relevant instances are ingnored This is the case starting around $C = 0.05$ for SVM (1) and around $C = 0.01$ for SVM (2).

Subtask e)

Normalization of data changes the relative scale of different features such that their values are comparable across features. For our data, that initially consists of features with similar scale, normalization does not have a significant impact on accuracy. For data with significantly varying scales of different features, normalization might affect accuracy to a higher degree as otherwise some features might have a way to strong/weak impact simply due to their scale.

Task 6

Before the start of any subtask, the data is prepared. To that end, the one-hot-encoded training and test data sets are loaded. Then, the descriptive features and the target vectors for training and test data are extracted.

Subtask a)

We first define the MLP using `MLPClassifier()` specifying five hidden layers with five neurons each and `logistic` as activation function. We restrict the number of iterations to 2000. Then, we train the MLP on the training data and make a prediction for the target feature of the test data. `fit` is used to train the MLP and `perdict` for the prediction.

Afterwards, we use `accuracy_score` of `sklearn.metrics` to compute an accuracy of 0.39 for our prediction. Similarly, we get the confusion matrix for the prediction depicted in table 4 using `confusion_matrix` of `sklearn.metrics`. The accuracy indicates a low rate of correct classification. A reason for this is found in the confusion matrix. The MLP always predicts `medium` seemingly regardless of input.

	`low`	`medium`	`high`
`low`	0	131	0
`medium`	0	33	0
`high`	0	104	0

Table 4: Confusion matrix for the MLP with 5 hidden layers of 5 neurons each using logistic activation function.

Note, that the execution of our code is not deterministic. The resulting accuracy and confusion matrix may vary. We attribute this behaviour to the random initialization of weights during training. We decided to use this specific accuracy and confusion matrix as it seems to fit the overall goal of this task. Other initializations can by chance result in significantly better performance (higher accuracy).

We further investigate our MLP using probability estimates for the test data. To this end, we compute the mean and variance of these estimates for the predictions `low`, `medium`, and `high`. These values are depicted in table 5. We see that on average the MLP predicts `medium` with a low amount of confidence. This result fits to the confusion matrix in table 4. Especially, because the variance for all three classes is 0.00. This would once again indicate the behaviour that `medium` is always predicted seen in the confusion matrix. The probability estimates for the first five instances depicted in table 6 also shows this.

	`low`	`medium`	`high`
0	0.41	0.47	0.12
1	0.41	0.47	0.12
2	0.41	0.47	0.12
3	0.41	0.47	0.12
4	0.41	0.47	0.12

Table 6: Mean and variance of probability estimates for the MLP with 5 hidden layers of 5 neurons each using logistic activation function.

Subtask b)

In this task, we are asked to improve the MLP created in subtask (a) by potentially preprocessing the data and by changing the MLP. To this end, we tested the effect of preprocessing on this specific data set. Normalization of the data using `StandardScaler()` of `sklearn.preprocessing` did not seem to improve the accuracy. This behaviour fits with the results seen in task 5. Similarly, the restriction to fewer descriptive features did not lead to an improvement of the MLP. Thus, we decided not use any preprocessing on the data.

On the other hand, we did modify the parameters of the MLP. We did not change the amount of hidden layers, but did vary the number of neurons in them. The first hidden layer now has 15 neurons, the second 7, the third 16, the fourth 8, and the last hidden layer 16. We arrive at these numbers by iteratively changing them and checking the performance based on accuracy. We also change the activation function used in all layers to `relu` as this leads to an immediate increase in accuracy. The final change to the MLP is an increase of the number of iterations from 2000 to 10000. It is straight forward that more training time should result in a better MLP.

These adjustments result in an accuracy of 0.94 and the confusion matrix depicted in table 7.

Note, that this MLP has the non-deterministic behaviour mentioned for subtask (a). We chose this particular result, as it shows that our changes lead to an improvement of the accuracy compared to the MLP in subtask (a). During our tests, we did not find that this MLP performs worse than the worst case performance of the MLP from the previous subtask. It is however possible when training both MLP often enough that the MLP in the previous subtasks performs better than the current one. This only happens for bad instantiations of this MLP and very good instantiations of the previous one. We also did not find that the MLP in subtask (a) can perform better than the best performance of the MLP in this subtask.

	`low`	`medium`	`high`
`low`	129	1	1
`medium`	12	90	2
`high`	0	1	32

Table 7: Confusion matrix for the adjusted MLP.

Subtask c)

We first train a naive Bayesian classifier on the unscaled training data using `fit()`. We than predict the target feature for the test data using `predict()` and compute the accuracy using `accuracy_score()`. This results in an accuracy of 0.62 and the confusion matrix depicted in table 8. Based on the confusion matrix, this model seems to struggle with the differentiation of `low` and `medium` as well as `medium` and `high`.

After normalizing the training and test data using `StandardScaler()`, we train a naive Bayesian classifier on the scaled data. For this model the accuracy is 0.39, and we get the confusion matrix depicted in table 9. Based on the confusion matrix, this classifier always predicts `medium`. We thus see a similar problem to the classifier without normalization of data. The likely reason for the decreased performance of this second model is the reduced difference between descriptive features after normalization.

	`low`	`medium`	`high`
`low`	131	0	0
`medium`	82	1	21
`high`	0	0	33

Table 8: Confusion matrix for the naive Bayesian classifier without normalization of data.

	low	medium	high
low	0	131	0
medium	0	104	0
high	0	33	0

Table 9: Confusion matrix for the naive Bayesian classifier with normalization of data.