Portfolio Assignment 1: Airbnb

This first assignment's main goal was to look at data from Copenhagen's Airbnb accommodations and figure out which ones can be marked as 'cheap' and which ones as 'expensive'. It also concerned binning the data based on price and visualization of the accommodations on a map. In this assignment, we followed the instructions, step by step, as they were very detailed and specific in what needs to be done.

Data acquisition: We loaded the data from the provided csv file. We did try getting the latest data From Airbnb, but the name field was now generic, so we chose to go with the provided data instead.

Data preparation: We removed rows with missing values, rows with 0 zero reviews, cleaned the neighborhood names missing special Danish characters and made price to be a float, from a string. We created bins for prices, to categorize them.

Data visualization: We created word clouds of AirBnB names and host names, to see the most mentioned words. We also used histograms to check the distribution of values. Map plots with colors signalizing first the price bins, then the neighborhood was used, to visualize the geographic position of the accommodations. We also used boxplots where we grouped by neighborhoods, to visualize availability and price. A simple bar chart was used to show the most active hosts. Descriptive analysis gave us a deeper understanding of the data.

Pre-processing: We removed room types that did not have much representation in the data set and did one-hot encoding on the neighborhoods and room type. We added another column, expensiveness, that marks whether an accommodation's price is above or below the median.

Models: We split the data into train and test set 80-20, where the y was the expensiveness. For features, we chose to keep latitude, longitude and two room types. We created two models, Naïve Bayes and k-NN. The Naïve bayes model ended up very good at predicting the affordable ones, but bad at the expensive ones, with precisions 0.93 and 0.55, recall 0.22 and 0.98, respectively. The k-NN performed equally on cheap and expensive places, with precision 0.67 and 0.66 and same recall.

Conclusion: We would choose the Naïve Bayes model, as it performs so well on the affordable apartments. The reason for having lower accuracy for the expensive ones could be caused by high variance in data.

Portfolio Assignment 2: Exoplanets

This assignment is working with data from K2 Kepler mission, provided by the NASA Exoplanet Archive. The main goal is to develop a model that would classify celestial objects and determine whether they are an exoplanet.

Data acquisition: We read the data from the given csv file. We renamed the columns for better readability.

Data preparation: We started out with 49 features. We first removed columns that were empty or irrelevant (ids, names). Next, we removed all rows with missing values. Looking at the correlation matrix, we saw that there are many features that highly correlate with others – those were ones that had a value, lower value, and upper value. We removed all columns that were the lower or upper values, and another highly correlating feature, leaving us now with 22 features.

Data visualization: To get a better understanding of the data, we visualized the values using boxplots and bar plots. Using histograms, we checked for normal distribution of all the features.

Pre-processing: Viewing the histograms, we selected those features that needed transformations – those that were right-tailed or left-tailed. Those were transformed and scaled to be more Gaussian-like. We decided to keep the outliers, as removing them would remove too much data – especially those with only 0s and 1s as values – where none of those were true outliers.

Models: We split the data into training and test sets. We decided to use Grid Search with Cross validation, using K-neighbors classifier. We gave it a range of hyperparameters, n-neighbors and different types of metrics, to determine the best one. The best metric ended up being Manhattan, with 6 neighbors. The model performed well, with 0.93 accuracy and all other metrics in the 90s too. We plot the confusion matrix, together with the precision-recall and ROC curve, showing that the model is in fact performing well.

Conclusion: K-neighbors classifier proved to be a fitting model to use for this problem. Eliminating more than half of the initial features and performing needed data transformations ensured the model's performance in all the given metrics.

Portfolio Assignment 3: Optimization of long-term correction of wind data

The purpose of this assignment is to figure out whether regression models could be used to predict wind direction and wind speed, instead of the neural network-based model that Vestas is using now to generate LTC wind data. To try and reduce computational cost and time.

Data acquisition: Out of the two possible ones, we chose to work with Risoe dataset – that is the same one as Vestas used, so we had something to compare our results to.

Data preparation: We chose to work with data from 77 as it had more data. We removed end periods with missing values. We kept values close to 0 for lack of domain knowledge. We replaced missing values by averaged values from previous years. We chose to keep the outliers, because after plotting wind speed data into a histogram, the data followed Weibull distribution very well, as it should. We changed the sample rate, to be measured once per hour. As time summer-winter time switch was not visible, we subtracted 2 hours from the times to convert it to UTC. To fill in missing data from wind direction, we did linear interpolation. We resampled the mast data to be in interval of 1 hour, to fit the MESO data. For resampling wind, we used the mean of sin and cos to get the mean direction, as it is circular data. We chose to interpolate MESO data from 80 and 60 to 77, to fit the MAST data.

Data visualization: To visualize data, we used many box plots, histograms, and mostly line plots to display the evolution of data over a time series.

Pre-processing: Since linear regression works best with scaled data, we used min max scaler to keep the outliers as part of the scale, so that the Weibull distribution is kept. We introduced a month-based feature to ensure that the test data is distributed over all seasons.

Models: We split data into train and test sets, stratified random sampling based on months. We created 3 models: Ordinary least squares, Elastic net with CV and kNN. For Elastic net and kNN, we gave it parameter grid to decide for the best ones. The best performing model was kNN for wind speed with MSE only 2.54 and R2 of 0.73. The other 2 models were trained in both wind direction and speed together, kNN separately. When averaging the MSE of kNN for wind speed and direction, it compares to the other models.

Conclusion: If only predicting the wind speed, regression models could be a possible option, however they would not be good enough for predicting wind direction, any of them.

Portfolio Assignment 4: The candidates

This assignment studied data from candidates for election of different Danish parties. Apart from analysis, the task was to develop models to classify people into parties, and another to itself split people into parties.

Data acquisition: Out of all datasets, we almost exclusively worked with alldata.xlsx, only using the separate ones to find the original questions, and the elected data for the last part.

Data preparation: We removed all Losganger, as they do not belong to any party. We looked at the correlation matrix of all the questions and saw that many highly correlate. Step by step, we dropped features with high correlation, leaving only those with 0.7 or lower, leaving us with 30 out of 50 questions.

Data visualization: We created a dictionary of color associated with each party to be used in plots. We did PCA to figure out the most crucial questions. We ran this multiple times after different steps of dropping correlating features. Though the two most important questions themselves changed, the topics remained the same – economics and immigration. Those two were used in plot of all politicians on a "political compass" map. For each question, we displayed the average answer per party, separately for DR and TV2 data. We also compared the average age of each party, ignoring candidates without age. Looking at the most extremist candidates, many from the top 20 were from the same parties. Looking at differences in responses between candidates, we used Jaccard distance between candidates' answers. The higher the percentage, the more different they are.

Pre-processing: We ran standard scaler on the data for pre-processing.

Models: For classification of who belongs to what party, we ran Decision tree, Random Forest, and Gradient boosted tree models, all with parameter grids to find the best hyperparameters. They performed gradually as mentioned, with GBT being the best. It did not misplace more than 3 members in any party and its total accuracy was 0.9. That is a very good result, considering that parties only had between 3 and 16 members, giving a limited data set to learn from. Though it also took the longest to run, almost 17 hours. For clustering analysis, we ran K-means, Hierarchical clustering and DBSCAN. The first two had a similar, quite successful result, suggesting possibility of fewer parties, as many overlap in the original dataset.

Conclusion: Looking at the overview of elected candidates compared to all of them, those that were in the middle (indifferent opinions) or in areas overlapping with other parties mostly did not get elected. Two parties had nobody elected, one in the middle and one completely overlapping.

Portfolio Assignment 5: Sentiment analysis

Assignment 5 concerns text analysis, where we developed a model analyzing whether a review of a movie is positive or negative, using bag of words.

Data acquisition: We loaded the data from the txt documents, where the reviews were used for features, and labels as prediction target.

Data preparation: We changed the labels from being a string 'positive' or 'negative' into binary values. As data was scraped from HTML, we removed the remaining HTML elements – 'br', breaking a line. We used CountVectorizer with 10000 features to put all the reviews as collections of the individual words, ignoring their order, purely looking at the count in each review. As we have 25000 reviews, we chose to work with 10000 most common words, out of the total 73925 different words used in all the reviews. As manually going through all the reviews and checking for spelling errors and correcting them is not suitable, setting the number of features to a lower number takes care of not including words occurring few times, such as typos.

Data visualization: We only used tables and simple printouts to visualize the data. We looked at the most used words, with leading articles, prepositions, and simple verbs.

Pre-processing: As all the data was only of values 0 and 1 in many columns, we did not do any pre-processing. Scaling or normalization is not in place for this type of data.

Models: We split the data into training, validation, and test data, in ratio 64, 16 and 20. We trained a simple neural network with one single hidden layer with 64 neurons, one dropout layer of 0.5 and a sigmoid output layer. We used kerner regularizer Ridge to try and prevent the model from overfitting by driving the weights low. Because it is a classification problem, we used binary crossentropy and a low learning rate to try and prevent overfitting. We ended up after trying many different parameters with those giving out the best model for this data. The model is running on the accuracy metric (validation loss might have been better). To further prevent overfitting, we included early stopping with patience of 50.

Conclusion: After training the model on train and validation data, we managed to get accuracy of 0.87 on test data. When trying out our own sentences, all except for one got classified correctly. From a human perspective, it gave those sentences with a "stronger" positive feeling a higher value. We mostly tested the difference of how it perceives 'and' and 'but' – in all cases the sentences that contained 'but' were marked as more negative than those with 'and' which does make sense.

Portfolio Assignment 6: Speech recognition

The main goal was to develop a model for speech recognition – recognizing four words from each other: yes, no, stop and go.

Data acquisition: The data is spectrograms which we are working with as matrices.

Data preparation: If we needed more data, we could have taken some samples and changed them – introducing more noise by increasing the brightness/sensitivity or moving the wave up or down to imitate differently pitched voices.

Data visualization: We printed some random samples of the spectrograms to see the visualized data. It is visible that they have been recorded by different people with different devices – resulting in different volume levels including noise and different heights of voice.

Pre-processing: We did not do any pre-processing, as the data we got was in the form of spectrograms.

Models: We split the data into training, validation, and test sets. First, we created a convolutional neural network model with 10 layers. Two layers are convolutional to create feature maps, after each of those there is a max pooling and a dropout layer. These are there to limit the feature map and strengthen the model to prevent overfitting. Then, a flattening layer prepares it for the next dense layer by making it into a vector. Lastly, another dropout layer is added for further regularization before the last and final dense output layer with a softmax function used for multiclass classification – we have 4 different words, so the output layer has 4 output nodes. We set the learning rate low to avoid overfitting and overshooting. We also defined early stopping with only patience of 3 due to a low number of epochs in the model, 20. We put it this low, because when we ran it the first time with higher epochs, it was overfitting highly. Our metric is validation loss. After training on train and val, the test data accuracy was 0.93 with 0.20 loss.

The other model we made was a simple NN with flattened layer, a dropout, and a dense layer with softmax. The test accuracy of this model is only 0.76. The model is not robust enough to extract the individual features. Same as CNN, it struggles the most with ‘No’ and ‘Go’ but significantly more, as it is not complex enough to tell small nuances apart.

Conclusion: CNN model is good for spectrograms, speech recognition, as the data is represented in a similar way as an image is, CNN is good at extracting key features from those. The hardest part for the model is telling difference between ‘No’ and ‘Go’ which is to be expected – this can be seen in the confusion matrix.

Final group project

For our final project, we chose to work with a data set concerning LEGO sets. The data is from two different APIs from Bricklink.com and Brickset.com. The main purpose of the project is to create a model that would estimate future prices of LEGO sets. This could be used for investment purposes, as some LEGO sets' current value is thousands of dollars more than the original set. The current prices are taken from a website where people re-sell their sets.

Data acquisition: We found and downloaded the dataset from Kaggle. The data contains information about LEGO sets from 1975 till 2023, including categorical, descriptive and price information.

Data preparation and visualization: First, as the main goal of the assignment is to estimate the current price, we looked at how much data there is for USD_MSRP (= Manufacturer's suggested retail price in USD). We saw that before 2005, there is almost no data on the original price, and only from 2007 the amount of data is relatively stable, covering approx. 2/3 of sets from the given year. That is why, onwards, we only kept data after 2005. To look at price increase over time, we looked at box plots where we grouped it by year – we saw that the median price per set slightly increases over time.

Next, we looked at the difference between USD_MSRP and the current price, grouping it by themes and subthemes. We created bar plots to see the difference both in dollars, and in percentage of the original value. There was a big difference between those, as some of the most expensive sets sold now are mostly Advanced models or Star Wars sets, however the percentage increase was highest by Minifigures.

As we are working with price over time, and the original data set has the USD_MSRP in whatever year the set is from, but the current price is from 2023, we needed to adjust the data for inflation. We used the CPI (= Customer purchase index) library to perform the adjustments. All the prices have been adjusted to 2022, as there is no data for the whole year of 2023, as it is not over yet. We compared the original to the inflated prices for all mentioned above.

Pre-processing: We removed rows that were missing values for either USD_MSRP or Current price. Together with those, we also removed all minifigures, as there were not enough, and they are outliers in the original price. We replaced all the NaN values in Minifigures column by 0.

We removed the column Subtheme, as there were too many, with too little in them. We also removed Packaging and Category, as both had one very dominant option and others with too little representation. We removed ID and Name, as they do not add value. From Availability, we dropped rows whose values were represented by less than 10 or were not specified.

Feature engineering: Instead of the year of release, we replaced this column by years since release. That showed that the oldest sets on average have a higher current value. We did one-hot encoding on availability, theme groups and themes.

Looking at correlation matrix, we chose to remove Pieces and inflation adjusted original price (both correlating with USD_MSRP). Them, we removed all the theme groups that highly correlated with themes.

Models: As we wanted to cover multiple different regression models, we decided to go with Linear regression, Elastic net, and Neural network. For each, we did model-specific pre-processing that best fits the given model.

We used MinMaxScaler for the Linear regression. We split it into training, validation, and testing sets. The model performed very poorly, R squared ended up negative, meaning that the model performed worse than a horizontal line. Linear regression does not do well with too many features (we had 102), that is why we tried to run it again, but removing themes and theme groups, leaving us with 9 features. That model performed acceptable for social “sciences”, with R squared of 0.6. When trying to train it on both train and validation data, it performed worse than only training data, with R squared of 0.52. This is possibly caused by the random state having more outliers in the test set (all models used the same split).

Elastic net, we used standard scaler for pre-processing. We used all features (with themes) and used Grid Search CV to find the best hyperparameters. It only chose to use Lasso, to drive the useless features to close to zero. Looking back at the correlation matrix, those were probably most of the themes, as they correlated very low with the current price (label). It performed with R squared of 0.7 on train and validation and of 0.65 on test data.

Neural network also used standard scaler and was trained with themes. We used a very low learning rate of 0.001 to prevent overshooting and overfitting. We had two dense layers and two dropout layers, one after each. For kernel regularizer, we got inspired by the result of the elastic net, choosing Lasso over Ridge, that improved the model a lot, together with increasing the factor. We did not use an activation function on the last layer since we are doing regression. This time, training + validation compared to test set performed comparably, with R squared of 0.75 and 0.74.

Conclusion: Out of the three models that we tried, the neural network not only performed the best, but also had the least difference of performance between the training + validation and test sets. Overall, different steps could have been taken during the pre-processing, removing features very lowly correlating with the label. Another possibility could have been to use bins for current prices, making this a classification problem instead of a regression problem.

To conclude, the result obtained by simple neural network is adequate.