# Homework 2 – MLOps (Fortnite with Hive)

*This document contains information regarding the homework given by the teachers in the PDP (Parallel Distributed Processing) course. For my homework, I was given the diabetes dataset for which I had to apply ML techniques, such as pre-processing, feature design and model assessment.*

## General information

Name: Alexander Arkhipov

Student number: 647833

GitHub URL: <https://github.com/paxer2k/pdp-assignments>

## Explanation of steps

1. Firstly, I created a jupyter notebook file and imported the diabetes dataset.
2. I then did a bit of research on the dataset, such as origin and meaning of every column in the dataset.
3. Furthermore, I performed every step mentioned in the document, such as pre-processing the data (removing outliers, normalizing data, encoding categorical variables), feature design (based on predictive features making new features) and applying a machine learning algorithm (applying a regression algorithm and assessing the score/accuracy of it based on the dataset)
4. Moreover, I also performed regression from a single data point to identify the spread of the disease one year after that datapoint (prediction)
5. I did further research into pipelines and how they work since this was a completely new topic for me.
6. I applied the pipeline to all of the steps mentioned in 3, except for displaying the score/accuracy, in order to keep the steps more structured, efficient and to automate the process.

## Explanation behind ML choices

### Pre-processing

**Explanation**

My pre-processing steps for this dataset consist of two main components, namely: processing the **numerical** and **categorical** variables from the dataset.

For the numerical data, I have used two steps: an **imputer** and a **scaler**. The reasoning behind the imputer is mainly due to the fact that it handles missing values in numerical features. The imputer will replace missing values with the median value of the corresponding feature. This is primarily to ensure that the pipeline can process complete data and that it won’t have to deal with any outliers. Despite me not having any missing values, I still think it is good practice to apply an imputer in the pipeline for numerical features. The reasoning behind using the scaler is that I wanted to normalize the numerical features by bringing them closer to a similar scale which can essentially help with various algorithms that are sensitive to the scale of the input features. This is especially important for regression algorithms.

For the categorical data, I have used two steps: an **encoder** and a **selector**. The reasoning behind the encoder is because I had to encode categorical features into a numerical representation because a large number of classifiers and algorithms cannot handle categorical variables and require numerical inputs. I chose OneHotEncoder because it is the most known way of encoding categorical variables by converting it into a binary vector format where every category becomes its own column, and the presence of the variable is determined by 0s and 1s. I chose to ignore the handling of unknown categories so that they do not raise an unexpected error. The reasoning behind the selector is mainly because I wanted to pick the most informative features based on a set of tests that this selector performs. The selector chosen was chi2 because it is good at measuring dependence between all of the categorical variables. From here, I chose to set the percentile parameter to 50 to only pick 50% of the most informative features from the selection. Even though the selector might not have been necessary since my dataset is relatively small and this is more applicable to larger datasets where dimensionality of the data matters, I still feel like this is a good practice as a whole. After all, there is a chance that this can improve the overall performance of a model and avoid issues such as overfitting.

### Feature design

**Explanation**

For my dataset I created two new features, which are **bmi category** (body mass index) and **bp category** (blood pressure).

The decision behind these two features primarily comes from the assessment of measurements of the top 3 predictive features according to different methods of measuring predictiveness. I have various methods of measuring predictiveness, such as feature importance, correlation matrix, univariate feature selection and mutual information scores. The most occurring that had the highest scores were: **bmi**, **s5** and **bp**. This means that they had the highest correlation with the dependent variable (the one we are trying to predict) which is **progression** of the disease.

Since these three features have the highest predictiveness, I thought it would be appropriate to make new features that are related to these three, hence why I came up with the bmi category and bp category.

The calculations behind these two features are relatively simple as they are related to the previously mentioned features. For the bmi category, I chose to go with the feature names of: ‘Underweight, ‘Normal Weight’, ‘Overweight’ and ‘Obese’ for which the bins (values) are: 0, 18.5, 25, 30, 100. Each category falls between the current number and the number before it (e.g, 0-18.5 -> ‘Underweight’). For the bp category, I went with a similar approach: ‘Normal’, ‘Elevated’, ‘High’, ‘Extremely High’ for the feature names and: 0, 80, 90, 120, 1000 for the bins (values).

Despite **bmi** and **bp** being two of the most predictive features, I still chose to go with my two created features (**bmi category** and **bp category**) to test whether I was going to get better results. Funnily enough, they increased the accuracy of the model (R-Squared) by **0.02** which is not a lot, but it is still an increase.

### Model selection

**Explanation**

For my dataset, I have chosen **regression** as my machine learning technique. The decision behind this choice lies with my dataset as well as the use case that I was given. I am supposed to predict **progression**, which is a numerical value, based on independent variables (features) which can be numerical or categorical.

In terms of the regression model, I have played around with different models and assessed them based on a few criteria. The criteria are related to the **execution time** of the pipeline and the score/accuracy of the model using **different formats**. The formats that were used to test the score/accuracy of the model are MSE (Mean Squared Error), MAE (Mean Absolute Error), Pearson’s correlation coefficient and R-Squared (determination coefficient).

For MSE and MAE, the lower the values are, the better the performance of the model since we are trying to lower the error rate. For Pearson’s correlation and R-Squared, the higher the values are, the better the performance of the model because we need to look for a high correlation and/or variance between the features (X and y).

I have chosen **LinearRegression** model because it has one of the least of time to execute from the majority of the available models, which is: **0.01 seconds**. On top of this, it has also had one of the best scores/accuracies for the previously mentioned formats. Quite a few other models, such as DecisionTreeRegressor, KNeighborsRegressor and SVR also have a low execution time, however, their MSE and MAE are much higher and Pearson’s correlation and R-Square much lower than LinearRegression. All in all, LinearRegression has the best results in both categories which makes it a good fit for this use case.

## Discussion of the answers to the assignment

For the assignment, I chose to take a different approach when it comes to the order in which I did the tasks. The order of the tasks for my assignment is as followed:

1. Find the top 3 predictive features according to 3 different methods of measuring predictiveness.
2. Design 2 new features.
3. Pre-process necessary features.
4. Predict how disease progresses 1 year from datapoint.
5. Report score/accuracy in at least 2 different formats.

I first started the assignment off by importing the necessary started libraries which I will need in the long run, which are: numpy, pandas, matplotlib, sklearn, and seaborn. After this, I stored all of the columns and values from the tab-separated-value file and renamed all of the columns to a lower-case because of personal preference.

1. For this portion, I used four methods for assessing the top 3 predictive features. The methods that I used were **feature importance, correlation matrix, univariate feature selection** and **mutual information scores.** The top 3 features based on these four methods appeared to be: **bmi**, **s5** and **bp**.
2. For this portion, I designed two new features based on point 1 since those features had the highest correlation to the variable we are trying to predict. The features that I implemented were **bmi category** and **bp category.** The values behind this are explained in the feature design portion.
3. For this portion, I chose to pre-process **numerical** and **categorical** variables. For the numerical variables I went with an **imputer** and a **scaler** to remove outliers/missing values and to standardize the data by bringing the values closer to each other. For the categorical variables I went with an encoder and selector to encode categorical variables and select a percentile of the most informative features from the dataset which would essentially help the algorithms perform better in terms of accuracy and score. This is further explained in more detail in the pre-processing portion.
4. For this portion I used the previously mentioned pre-processing techniques and feature design variables to perform a **linear regression** analysis using the pipeline method to automate the process. For the X variable I chose **bmi category**, **bp category** and **s5** features and for the y variable I chose the **progression** of the disease. I split these variables into train and test sets and used a random data point to predict the progression of the disease (first row of the testing dataset) which gave me a value of **179**.
5. For this portion I used various methods to assess the quality of the algorithm by using different formats such as: **MSE**, **MAE**, **Pearson’s correlation coefficient** and **R-Squared**. These formats were performed on the previously pre-processed techniques along with the train and test splits. The tests itself happened on the testing data using the pipeline prediction method. The results for the linear regression model are as followed:

* Mean Squared Error (MSE): **2869**
* Mean Absolute Error (MAE): **43.6**
* Pearson’s correlation coefficient: **0.68 or 68%**
* Determination coefficient (R-Squared): **0.46 or 46%**

The choice behind these methods is explained in the model portion, however, the biggest reason is because they are the most used methods for assessing regression algorithms. For MSE and MAE, the lower the values are, the better the accuracy of the model. As for Pearson’s correlation coefficient and R-Squared, the higher the values are, the better the accuracy of the model.

## Time of pipeline execution

A picture containing text, font, line, screenshot

Description automatically generated

In case the screenshot is hard to read, the execution of the linear regression was **0.01 seconds**.

Figure Execution time of LinearRegression