ML02

Păpăluță Vasile Vladimir Stojoc Răzvan Fișer
Marius Purici Covalevschi Andreea Smocvin Denis
Graur Elena Balamatiuc Eduard Andronovici Darinela
Clefos Alexandru

November 7, 2022

Contents

1 Statistical Analysis. Pandas

One of the main goals of classification machine learning models is to find patterns that will allow the algorithm to separate the classes of samples. Usually, they do that using different approaches. People can do that too, and as a data scientist, it is essential to find manually the patterns that will allow a Machine Learning algorithm to do that. Usually, we use for that Statistical tools and methods.

1.1 The tasks:

• Create a new Jupiter notebook.
• Study each column of the selected Data Set
  We highly recommend exploring the following statistical parameters:
  - Mean.
  - Median.
  - Mode.
  - Standard Deviation.
  - Variation.
  - Range of values.

• Find what columns have the highest absolute correlation with the **target** column (target column is specified in the Data Set page), and explain why you think so happens. (here, you can explore the literature on the topic).

1.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Heart Disease (https://bit.ly/3Nkw8fd)
- Pima Indians Diabetes Database (https://bit.ly/3zqVZMs)
- Titanic - Machine Learning from Disaster (https://bit.ly/3sJhAfo)
- Wine Quality Data Set (https://bit.ly/3WgcSDt)

2 Numpy

Numpy is a powerful library that takes almost all computing tasks needed for Machine Learning, from vector and matrix operations to optimization problems. Usually, all formulas in every algorithm are implemented in NumPy or using another library named scipy. In this homework, we invite you to write some python functions using NumPy to practice more the maths to NumPy conversion, which will be helpful to you in the Research and Development projects:

2.1 The tasks:

2.1.1 Formula Nr. 1: Normal Distribution

The normal distribution function is the function that allows us to get the probability of getting a value from a numerical series knowing the mean and standard deviation of the series.

$$N(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{1}$$

Implement in python using NumPy the formula above as a function which will take the following parameters:

- x: a float value.
- μ (mu): a float value (represents the mean of the series).
- σ (sigma): a float value (represents the standard deviation of the series).

2.1.2 Formula Nr. 2: Sigmoid Function

The name of the organization comes from an activation function - Sigmoid. The sigmoid function is used when you need to convert real numbers into probabilities or from a -∞ to $+\infty$ range into a 0 to 1 range. Its formula is the following:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Implement in python using NumPy the formula above as a function which will take the following parameters:

- x: a real numeric value.

2.1.3 Formula Nr. 3: Weights update in Logistic Regression

One of the Machine Learning algorithms you will learn during this book is Logistic Regression. It uses an iterative process to find the coefficients of a linear function, updating the values of the weights vector (w) at each iteration, taking into account errors of the model's prediction itself.

$$w_j = w_j - \alpha \frac{1}{n}(\hat{y}_i - y_i)x_j^{(i)} \tag{3}$$

In the formula above, w_j is the weight with the index j in the weight vector, y_i is the value with the index i in the target vector, while $x_j^{(i)}$ is the value of the matrix in the cell with the row index i and the column index j. n is the number of rows in the matrix X and vector y, α is the learning rate.

Implement in python using NumPy the formula above as a function which will take the following parameters:

- w (weight vector): a 1-d numpy array.
- X (matrix of samples): a 2-d numpy array.
- y (target value vector): a 1-d numpy array.
- $\hat{y}$ (predicted target vector): a 1-d numpy array.
- α (learning rate): a value from 0 to 1, default - 0.0005.

The function should iterate through rows of X and y, and update the vector w. The $\hat{y}$ can be a random vector of numbers in range from 0 to 1. Return the newly updated w array.

2.1.4 Formula Nr. 4: Mean Squared Error

Mean Squared Error(MSE) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. They are usually used in Regression problems.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4}$$

Why square the difference between the true value and the predicted one? Because the errors can be positive and also negative, and as you might guess, they can cancel one another, which is not good.

You might ask, why not use the absolute value to not cancel values? Well there is a bit complicated answer for which you have to understand the concept of gradient descent, so we will leave it for your research.

In the formula above, y represents the predictions vector, for example, predictions from some ML algorithm, and $\hat{y}_i$ represents the actual target values, which we try to predict. $\sum$ sign means just a summation operation, where $i = 1$ represents the initial value of the index and n represents the length of the vector. For every i till n, you have to calculate the squared difference for the predicted and target value for row i, sum all this together, and then calculate its mean by dividing by n.

Implement in python using NumPy the formula above as a function which will take the following parameters:

- y (target): a 1-d numpy array.
- $\hat{y}$ (predictions): a 1-d numpy array.

The function should return the error

2.1.5 Formula Nr. 5: Binary Cross Entropy

This is another error function that is used in Classification problems, the y parameter from the function below represents the actual value of the class and $\hat{y}$ the predicted one. You might observe that always half of the function is canceled, which part depends on the actual class. If $y = 0$, the first half is canceled, and the $(1 - y)$ will be 1, which saves the second part, for $y = 1$ the opposite. The *log* is taken out of the predicted value, and as the value approaches the actual value, this *log* error will be smaller and smaller, showing ML algorithms the way to the best parameters.

$$Loss = -\frac{1}{output\_size} \sum_{i=1}^{output\_size} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i) \tag{5}$$

In the formula above, y represents the predictions vector, for example, predictions from some ML algorithm, and $\hat{y}$ represents the actual target values, which we try to predict. $\sum$ sign represents just a summation operation, where $i = 1$ represents the initial value of the index and *output_size* represents the length of the vector, for every i till *output_size* you have to calculate the error for predicted and target value for row i, sum all this together and then calculate its mean by dividing by *output_size*.

Implement in python using NumPy the formula above as a function which will take the following parameters:

- y (target): a 1-d numpy array.

- $\hat{y}$ (predictions): a 1-d numpy array.

The function should return the error

3 Linear Regression

In the majority of Machine Learning use-cases, following one or another form as a Data Scientist, you should create a model that predicts something. Be it a class, a numerical value, or a sequence of values. In this task, you will train your first Machine Learning Model - **Linear Regression**.

3.1 The tasks:

In this task, we invite you to train 2 models on the selected Data Set that should predict the target column. The models are the following:

- the LinearRegression from sklearn.
- the Lin_reg implementation offered in SMLH.

The tasks:

- Create a Jupyter notebook with a clean code.
- Study the correlation between features, find the features subset with the highest correlation with the target column, and try to explain from the business point of view why they have such a big correlation.
- Create a second set of data with the columns that have an absolute correlation between 0.5 and 0.8 with the target column.
- Split the data into 2 sub-sets using the train_test_split function from sklearn.
- Train a sklearn Linear Regression model on the data provided to you.
- Train a from-scratch implementation of Linear Regression on the train sub-set.
- Test the models on the test sets from the initial data set. For error metrics, use the model's 'score' function from the Linear Regression model.
- Split the data with the selected columns into 2 sub-sets using the train_test_split function from sklearn.
- Train a sklearn Linear Regression model on the data with selected columns (train subset).
- Train a from-scratch implementation of Linear Regression on the train sub-set.
- Test the models on the test sets from the initial data set. For error metrics, use the model's 'score' function from the sklearn Linear Regression model.

- Please try to interpret the results you are getting by comparing the error of the models you created.
- Please comment on your code.

3.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Boston Houses Prices (https://bit.ly/3t4nb0h)
- Used Car Data (https://bit.ly/3sLi4lj)
- Electric Motor Temperature (https://bit.ly/3UuvaPV)

4 Logistic Regression

In the previous topic, you learned about regression tasks in the Machine Learning field. Classification is another popular type of task in Machine Learning, where instead of learning to predict a continuous value, you learn the predict the discrete ones. The first type of classification model that you will learn will be the **LogisticRegression**, which uses linear regression as a base model and adapts it to classification tasks.

In this task, we invite you to train 2 models on chosen Data Set that should predict the class column. The models are the following:

• the LogisticRegression from sklearn.

• the Log_reg implementation offered in SMLH. (Sigmoid Machine Learning Handbook)

4.1 The tasks:

For this chapter, you will have to do the following:

• Create a Jupyter notebook with a clean code.

• If your classes are expressed as non-numerical features, map them to numbers, for example, 0, 1, 2, etc.

• Study the correlation between features, find the features subset with the highest correlation with the target column, and try to explain from the business point of view why they have such a big correlation.

• Create a second set of data with the columns that have an absolute correlation between 0.5 and 0.8 with the target column.

• Split the data into 2 sub-sets using the train_test_split function from sklearn.

• Train a sklearn Logistic Regression model on the data provided to you.

• Train a from-scratch implementation of Logistic Regression on the train sub-set.

• Test the models on the test sets from the initial data set. For error, metric use the accuracy_score function from sklearn.metrics.

• Split the data with the selected columns into 2 sub-sets using the train_test_split function from sklearn.

• Train a sklearn Logistic Regression model on the data with selected columns (train subset).

• Train a from-scratch implementation of Logistic Regression on the train sub-set.

• Test the models on the test sets from the initial data set. For error, metric use the accuracy_score function from sklearn.metrics.

• Please try to interpret the results you are getting by comparing the model's accuracy on the test sets.

• Please comment on your code.

4.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Heart Disease (https://bit.ly/3Nkw8fd)
- Pima Indians Diabetes Database (https://bit.ly/3zqVZMs)
- Titanic - Machine Learning from Disaster (https://bit.ly/3sJhAfo)
- Wine Quality Data Set (https://bit.ly/3WgcSDt)
- Body Performance Data (https://bit.ly/3FBtYWu)
- Heart Failure Prediction Dataset (https://bit.ly/3TR0yYU)
- Telco Customer Churn (https://bit.ly/3fqDwc7)

5 Principal Component Analysis

Till now, you learned 2 algorithms for supervised Machine Learning. However, in the real world, labeled data is very expensive. It often costs a lot for a company to get labeled data, and they are asked to work with the unlabeled one and extract hints and insights from it. In this task, we invite you to explore one of the types of unsupervised learning - **Dimensional Reduction**.

This task is made up of 2 subtasks:

5.1 Subtask 1:

In the first, you are going to work with the Iris Data Set.

• Load the Data Set (https://bit.ly/3WiUJ7W)

• Plot the first 3 columns of the Data Set and the target column in a 3d scatter plot.

HINT: Use the target column as a hue for the points.

LINK: (https://bit.ly/3SRIsEJ)

• Copy the implementation of the Principal Component Analysis algorithm from SMLH.

• Create an instance of the Scratch-made PCA algorithm using the eigenvector algorithm.

• Create an instance of the Scratch-made PCA algorithm using the SVD algorithm.

• Create an instance of sklearn and implement a PCA algorithm.

• Apply StandardScaler from sklearn on the data.

• Train all created instances of PCA on the iris data.

• Use each created instance of PCA to reduce the dimensionality of the Data Set to 2, creating in such a way 3 separated Data Sets.

• Create a plot that combines 3 scatter plots and plots each of the result Data Sets in a separate window. Each window should be related to which version of the PCA was used.

HINT: (https://bit.ly/3DIzAwW)

• Try to conclude what you see by comparing the results of each algorithm and the 3d scatter plot you made before.

• Retrain a sklearn PCA model on the iris Data Set with *n_components* set to 1.

• Create a line plot of the *explained_variance_ratio*.

• Try to make a conclusion based on that plot.

5.2 Subtask 2:

In the second task, you will work with another Data Set - images of digits:

- First, load the Data Set: (https://bit.ly/3SOwFqL)
- Second plot in a multiple window plot 10 random digits with a title having the digit from the image on a 2 x 5 grid.
- Create an instance of the sklearn PCA.
- Apply scaling on the digit Data Set.
- Train the PCA on the digit Data Set.
- Reduce the dimensionality of the Data Set to 2 dimensions.
- Plot the new result Data Set in a scatter plot with each point colored by its digit label.

6 Probability. Naive Bayes

In this task, you are going to explore the next Classification Machine Learning Algorithm - **Naive Bayes**. There are actually more than one type of this algorithms, so you will get to test three of them. Naive Bayes is a ML model that is used for data of large volumes. One of the recommended algorithms used in these scenarios.

6.1 The tasks:

For this chapter, you will have to do the following:

- Load the selected Data Set.
- Split the data set into the train and test set.
- Analyze and Preprocess your data
- Create an instance of Gaussian Naive Bayes (https://bit.ly/3DTv0fu)
- Create an instance of Multinomial Naive Bayes (https://bit.ly/3sOP4Jn)
- Create an instance of Bernoulli Naive Bayes (https://bit.ly/3WkuC0t)
- Train every version of Naive Bayes on the train data set and make predictions on the corresponding test subsets.
- Get the accuracy of model.
- Create a table of the following structure as a pandas data frame.

ML Model	Accuracy

- Make a conclusion based on the data frame that you got.

6.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Telco Customer Churn (https://bit.ly/3fqDwc7)

7 Decision Tree

As you probably already learned, **Decision Trees** are very powerful and, at the same time, easy-to-understand Machine Learning algorithms. Doing this homework, you will train a couple of Decision tree algorithms and will try to dive deeper into how it works.

7.1 The tasks:

In this task, we invite you to train 2 models on a selected Data Set that should predict the class column. The models are the following:

• Let's, of course, start with importing the Data Set.

• The next thing you will do is create a visualization of the algorithm's decision boundaries to understand better how it works.

- Choose 2 features of the Data Set with the highest absolute correlation with the target.
- Create the decision boundary graph of these features related to the target.
- Comment on the result that you got.

PS: More about decision boundary you can find here: (https://bit.ly/3U7ICc7)

• Decision trees have a lot of hyperparameters. Let's explore how they influence the accuracy of the model. For every hyperparameter in this list, create a line plot where on the X-axis are hyperparameter values and on the Y-axis is the accuracy of the model for different hyperparameter settings:

- *max_depth*
- *min_samples_split*
- *min_samples_leaf*
- *min_weight_fraction_leaf*
- *max_features*
- *max_leaf_nodes*
- *min_impurity_decrease*

P.S. Always set *random_state* the same.

• Comment on the impact of every hyperparameter on the model's accuracy.

• Train a model with the best set of hyperparameters, then export the tree's structure.

7.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Heart Disease (https://bit.ly/3Nkw8fd)
- Pima Indians Diabetes Database (https://bit.ly/3zqVZMs)
- Titanic - Machine Learning from Disaster (https://bit.ly/3sJhAfo)
- Wine Quality Data Set (https://bit.ly/3WgcSDt)
- Boston Houses Prices (https://bit.ly/3t4nb0h)
- Avocado Prices (https://bit.ly/3UenS2A)

8 Random Forest

Now that you learned how Decision Trees work, it's time to move forward to the next level - **Random Forest**. In this homework, you will explore Random Forest, comparing it with the result from Decision Trees.

You may use the same data set you used in the last homework.

8.1 The tasks:

In this task, we invite you to train 2 models on the selected Data Set that should predict the class column. The models are the following:

• Let's start with importing the Data Set.

• The next thing you are going to do is to create a visualization of the decision boundaries of the algorithm, to better understand how it works and compare it with the one of the Decision Tree.

- Choose the same 2 features from the last homework.
- Create the decision boundary graph of these features related to the target.
- Comment on the result that you got, and compare them with the one from Decision Trees homework.

PS: More about decision boundary you can find here: (https://bit.ly/3U7ICc7)

• Random Forest, as Decision trees, has a lot of hyperparameters. Let's explore how they influence the accuracy of the model. For every hyperparameter in this list, create a line plot where on the X-axis are hyperparameter values and on the Y-axis is the accuracy of the model for different hyperparameter settings:

- *max_depth* - *min_samples_split*
- *min_samples_leaf*
- *min_weight_fraction_leaf*
- *max_features*
- *max_leaf_nodes*
- *min_impurity_decrease*

P.S. Always set *random_state* the same.

• Comment on the impact of every hyperparameter on the model's accuracy, especially comparing the Decision Trees results.

• Train a model with the best set of hyperparameters.

8.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Heart Disease (https://bit.ly/3Nkw8fd)
- Pima Indians Diabetes Database (https://bit.ly/3zqVZMs)
- Titanic - Machine Learning from Disaster (https://bit.ly/3sJhAfo)
- Wine Quality Data Set (https://bit.ly/3WgcSDt)
- Boston Houses Prices (https://bit.ly/3t4nb0h)
- Avocado Prices (https://bit.ly/3UenS2A)

9 Support Vector Machines

Support Vector Machines (SVMs) is one of the most powerful machine learning models. On small amounts of data, it may even outperform Neural Networks. So let's explore what its power consists of.

9.1 The tasks:

For this chapter, you will have to do the following:

• Import the Data Set.

• Firstly, we will explore how the feature scale influences the algorithm's performance. Train an SVM model on 4 versions of the Data Set:

- Initial one.
- The one passed through the StandardScaler.
- The one passed through the MinMaxScaler.
- The one passed through the MaxAbsScaler.

Create the following table in pandas and compare the results. Comment them:

Initial	standard_scaler	min_max_scaler	max_abs_scaler

• Usually, SVMs use some Kernel to move the data to another space where linear planes separate the classes. In this task, you will have to explore how different kernels influence the decision boundary. Create the plots of decision boundary for the following kernels:

- linear
- poly
- rbf
- sigmoid

Plot them all together in a combined plot, compare them, and express some conclusions.

9.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Heart Disease (https://bit.ly/3Nkw8fd)
- Pima Indians Diabetes Database (https://bit.ly/3zqVZMs)
- Titanic - Machine Learning from Disaster (https://bit.ly/3sJhAfo)
- Wine Quality Data Set (https://bit.ly/3WgcSDt)
- Body Performance Data (https://bit.ly/3FBtYWu)
- Heart Failure Prediction Dataset (https://bit.ly/3TR0yYU)
- Telco Customer Churn (https://bit.ly/3fqDwc7)
- Chronic Kidney Disease dataset (https://bit.ly/3UejZKK)
- Digit Recognition (https://bit.ly/3SOwFqL)

10 Missing Values Imputation

As you learned from the last topic, data rarely comes in a structured way with all values filled. Usually, the data comes with missing values along the way, which you need to handle, and this is what you will do in this homework.

10.1 The tasks:

For this chapter, you will have to do the following:

- Let's begin by importing the Data Set.
- Create a (bar) plot showing the number of missing values in every column.
- Split the Data Set into the train and test sets.
- Create an instance of the SimpleImputer with the strategy set as mean. (https://bit.ly/3Wh7tvN)
- Create an instance of the SimpleImputer with the strategy set as **median**.
- Create an instance of the SimpleImputer with the strategy set as **most_frequent**.
- Create an instance of the SimpleImputer with the strategy set as **constant** and the **fill_value** = 0.
- Create an instance of the CDI from reparo.
- Create an instance of the FRNNI from reparo.
- Create an instance of the HotDeckImputation from reparo.
- Create an instance of the KNNImputer from reparo.
- Create an instance of the PMM from reparo.
- Create an instance of the SICE from reparo.
- Create an instance of the MICE from reparo.
- Train every imputer on the train Data Set.
- Fill in the missing values on the train and test subset.
- Train the following models on every set that you got after imputing the missing values:

    - Logistic Regression.
    - Gaussian Naive Bayes.
    - TreeDecisionClassifier.
    - RandomForest.

- Test each of the models on the test set and build the following table:

Imputation Algorithms	Prediction algorithms	Accuracy

- Conclude the results you got.

10.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Titanic - Machine Learning from Disaster (https://bit.ly/3sJhAfo)
- Chronic Kidney Disease dataset (https://bit.ly/3UejZKK)
- Boston Houses Prices (https://bit.ly/3t4nb0h)

11 Hyperparameter Tunning

In the last topic, you learned what **Hyperparameter Tuning** is. In this homework, we invite you to explore the effect of hyperparameters on the performance metrics of different models.

11.1 The tasks:

For this chapter, you will have to do the following:

- Import both Data Sets.
- Clean both Data Sets.
- Create a line plot with the number of neighbors of KNN on the X-axis and the model's accuracy on the Y-axis on the classification Data Set.
- Create a line plot with the number of neighbors of KNN (Regression version) on the X-axis and the MSE of the model on the Y-axis on the regression Data Set.
- Create a heatmap using the TreeDecissionClassifier using the **max_depth** as X-axis, and **max_features** as the Y-axis and the color of the map should depend on the accuracy of the model on the classification Data Set for the **criterion = gini**
- Create a heatmap using the TreeDecissionClassifier using the **max_depth** as X-axis, and **max_features** as Y-axis, and the color of the map should depend on the accuracy of the model on the classification Data Set for the **criterion = entropy**.
- Create a heatmap using the TreeDecissionRegression using the **max_depth** as X-axis, and **max_features** as Y-axis, and the color of the map should depend on the MSE of the model on the regression Data Set for the **criterion = friedman_mse**.
- Create a heatmap using the TreeDecissionRegression using the **max_depth** as X-axis, and **max_features** as Y-axis, and the color of the map should depend on the MSE of the model on the regression Data Set for the **criterion = poisson**.
- Create a line plot using the RandomForestClassifier containing 2 lines for **criterion = gini** and one for **criterion = entropy** on X-axis should be the **n_estimators** and one Y-axis the accuracy of the model on the classification Data Set.
- Create a line plot using the RandomForestRegressor containing 3 lines for **criterion = "squared_error"**, **criterion = "absolute_error"** and one for **criterion = "poisson"**, on X-axis should be the

n_estimators and one Y-axis the MSE of the model on the regression Data Set.

- Create a line plot using the SVC model containing 3 lines for **kernel= ’poly’, ’rbf’, ’sigmoid’** on the X-axis should be the **C** and one Y-axis the accuracy of the model on the classification Data Set.
- Create a line plot using the SVR model containing 3 lines for **kernel= ’poly’, ’rbf’, ’sigmoid’** on the X-axis should be the **C** and one Y-axis the MSE of the model on the regression Data Set.
- Please extract a conclusion for every plot that you make.

11.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Used Car Data (https://bit.ly/3sLi4lj)
- Electric Motor Temperature (https://bit.ly/3UuvaPV)
- Body Performance Data (https://bit.ly/3FBtYWu)

12 Gaussian Mixture Models. EM algorithm. KMeans

You already got to know the clustering problem in Machine Learning. Also, you met one of the business problems where these algorithms are used - customer segmentation. Today you are going to practice this knowledge on a very similar task.

12.1 The tasks:

For this chapter, you will have to do the following:

- Bring all columns into a numeric form: applying mapping or dummy variables.
- If you think it can be helpful, you may apply feature scaling.
- Using Silhouette score, find the best number of clusters for GMM (implemented in sklearn.)
- Using Silhouette score, find the best number of clusters for KMeans (implemented in sklearn.)
- Plot the process of choosing the best number of clusters for each algorithm and try to explain.
- Cluster the Data Set using both algorithms.
- Extract from the KMeans algorithm the centroids.
- Extract from GMM the means of the clusters.
- Build the following table for both algorithms, and fill the table with the means of the clusters.

	feature 1	feature 2	...	feature m
cluster 1				
cluster 2				
...				
cluster n				

- Make a conclusion based on the table you got and create an imaginary customer for every cluster.

12.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Customer Segmentation (https://bit.ly/3FG0Woy)

- Wine Customer Segmentation (https://bit.ly/3Nn242k)
- Customer Personality Analysis (https://bit.ly/3zw8Woz)

13 Data Visualization

Data visualization is the graphical representation of information and data. Using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. This is what you will do today.

13.1 The tasks:

For this chapter, you will have to do the following:

- Import the Data Set.
- Prepare the dataset.
- Repeat the plots that were presented in the book
- Search for new plots that you could build, which might come from different libraries. Here is just your imagination and curiosity.

13.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Heart Disease (https://bit.ly/3Nkw8fd)
- Pima Indians Diabetes Database (https://bit.ly/3zqVZMs)
- Titanic - Machine Learning from Disaster (https://bit.ly/3sJhAfo)
- Boston Houses Prices (https://bit.ly/3t4nb0h)

14 Outlier Detection

During the last topic, you learned the art of plotting using matplotlib and seaborn in python. Also, we have learned what **Outliers** are and how to identify them. In this topic, you should explore a little bit more the given Data Set and plot the information about the outliers.

14.1 The tasks:

For this chapter, you will have to do the following:

- Import and prepare the Data Set.
- Find the 2 columns that correlate most significantly with the price column.
- Create a temporary subset using these 2 columns.
- Train and predict using the Isolation Forest, OneClassSVM, Elliptic Envelope, and LocalOutlierFactor.
- Create a grid 2x2 on scatter plots with the different colors for normal samples and outliers for each algorithm.
- Try to make a conclusion based on these plots.
- Find the number of outliers on the whole Data Set (without the target column) for different values of **contamination** for each outliers detection algorithm.
- Create for each algorithm the line plot showing the dependence of the number of outliers and the contamination value.
- Try to conclude.
- Split the Data Set into the train and test sets.
- Using the default settings of each outliers detection algorithm, find and remove the outliers from the Data Set, creating a subset.
- Train on the initial and the gotten subset of the following algorithms: LinearRegression and KNN Regressor.
- Find the accuracy of each combination of the prediction and outliers detection algorithms on the test subset.
- Create the table of the following structure.
- Make a conclusion based on the table that you got

14.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Pokemon Classification (https://bit.ly/3sJpeqn)

15 Feature Engineering

Feature engineering refers to using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modeling.

15.1 The tasks:

For this chapter, you will have to do the following:

- Using Feature Engineering try the algorithms you discovered and apply them to the previous Data Sets.

16 Feature Selection

Feature Selection is reducing the input variable to your model by using only relevant data and eliminating noise in data.

16.1 The tasks:

For this chapter, you will have to do the following:

• Using Feature Selection try the algorithms you discovered and apply them to the previous Data Sets.

17 Class Balancing

In Machine Learning, **Class Balancing** means balancing a Data Set where number of classes are unbalanced. Avoiding Class Imbalance is essential before using a machine learning algorithm because our end goal is to train a machine learning model that generalizes well for all possible classes assuming we have a binary dataset with an equal number of samples.

17.1 The tasks:

For this chapter, you will have to do the following:

- Using Class Balancing, try the algorithms you discovered and apply them to the proposed Data Sets.

17.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Pokemon Classification (https://bit.ly/3sJpeqn)

18 Model Interpretation

In the last topic, you learned how to interpret a model. In this homework, we invite you to apply all the knowledge to interpret the models' results on the Data Sets.

18.1 The tasks:

For this chapter, you will have to do the following:

• Import both Data Sets.

• Train a linear regression model on the regression Data Set and make a bar plot using the weights of the linear model. Interpret the plot.

• Generate the classification Data. Set the image showing the decision tree.

• For both Data sets, after it was cleaned, split them into the train and test subsets.

• On the train subsets of the regression Data Set to trail the following algorithms: Linear Regression and KNN Regression (bonus: try RandomForestRegression and DecissionTreeRegression).

• On the train subsets of the regression Data Set to trail the following algorithms: Logistic Regression, Gaussian Naive Bayes, and KNN Clasifcator (bonus: try RandomForestClassification, DecissionTreeClassification, Bernoulli Naive Bayes, and Multinomial Naive Bayes).

• For each regression model, get the MSE, MAE, and RMSE on the test subset.

• For each classification model, get the Accuracy, Precision, Recall, and F1-Score.

• For each regression model, build the following table.

Estimator	MSE	MAE	RMSE
est 1			
est 2			
...			
est n			

• For each classification model, build the following table.

Estimator	Accuracy	Precision	Recall	F1-Score
est 1				
est 2				
...				
est n				

• Analyze the tables you got, make some conclusions, and choose the best regression and classification, models.

• Pick the best classification model and interpret it using shap: Create the following plots:

- SHAP barplot.
- SHAP waterfall plot.
- SHAP beeswarm plot.
- SHAP force plot.

• Make some conclusions on each plot.

• Pick the best regression model and interpret it using LIME. Find the samples with the highest difference between the target and predicted values. Interpret them with LIME and say what went wrong.

• Make some conclusions on each plot.

18.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

• Body Performance Data (https://bit.ly/3FBtYWu)
• Heart Failure Prediction Dataset (https://bit.ly/3TR0yYU)
• Telco Customer Churn (https://bit.ly/3fqDwc7)
• Used Car Data (https://bit.ly/3sLi4lj)
• Electric Motor Temperature (https://bit.ly/3UuvaPV)

19 Streamlit

As you probably already learned, **Streamlit** is one of the best tools to help you make your projects, based on machine learning models, more attractive and interactive. For this task, we encourage you to use online documentation when needed because that's the best way to understand all the tools this framework provides.

19.1 The tasks:

For this chapter, you will have to do the following:

- Create a home page with the description of the dataset you choose
- Add a sidebar where you will be able to switch between the created pages
- On the next page, add multiple input value widgets that will be used to change the parameters of your model
- Add a data display element that will print the current head of the dataset
- After changing the parameters, add a process button that will be used to start training a new model with the input parameters given.
- Add a container in which will be written all the parameters you set
- Below the container, add at least 3 charts that will print the correlation between the columns or the prediction for categories and others
- For each text element, use a different type (i.e. markdown, code block, title, header, LaTeX)

19.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- Heart Disease (https://bit.ly/3Nkw8fd)
- Pima Indians Diabetes Database (https://bit.ly/3zqVZMs)
- Titanic - Machine Learning from Disaster (https://bit.ly/3sJhAfo)
- Wine Quality Data Set (https://bit.ly/3WgcSDt)
- Heart Failure Prediction Dataset (https://bit.ly/3TR0yYU)

20 Memory optimization. Dask

The process of training a model can be challenging and both computation- and memory intensive. **Dask** is one of the solutions to handle the issue of memory limitations in the case of working with a large amount of data.

20.1 The tasks:

For this chapter, you will have to do the following:

- Import one of the datasets above as a Dask object and set the number of partitions;
- Prepare the dataset for further processing;
- Output the head of the current data and the description for each remained column;
- Select a criteria to group the data by and output the significant data for each group;
- Choose a suitable algorithm from the ones studied earlier and apply it to the analyzed dataset;
- Plot some significant data using Dask.

20.2 Possible Data Sets

Here is a list of Data Sets on which you can apply knowledge from this chapter:

- 15 Million Chess Games from Liches (2013-2014) - (https://bit.ly/3SoHujG)
- Ocular Disease Recognition - (https://bit.ly/3ffoEgn)
- Spanish Rail Tickets Pricing – Renfe (https://bit.ly/3Sah9pN)
- Airline Delay and Cancellation Data (2009 - 2018) - (https://bit.ly/3RbM0AV)