This file explains the main structure of the provided code and how to use it.

1) Data Acquisition

LOGIC:

The data Aqcuisition is handled by the Main Folder "Data Scraping". The General logic behind our Scraping is that we define a scraping Pipeline, which is executed within the User Terminals, for e.g. "Scraping Terminal Chris". This script runs the webscrape_loop_optimized from "Data Scraping/Functions/Pipeline_Functions.ipynb". The webscrape loop is provided with User Information, such that it knows where (on the cloud) to store the scraped data and what exactly has to be logged to the "logging/logging_dfs/logging_df_user", which controls that whenever you Restart the scraping process, the next-in-line car observations for that user are scraped. This makes sure, that we do not scrape the same offers repeatedly and that you can stop and Restart the scraping process whenever you like in a convenient manner. Since we scrape data simultaneously over 4 different Computers, the logging_df also makes sure that the scraping Tasks are distributed evenly to the different Computers, which are identified by the "user" variable in the Scraping Terminals.

USAGE:

1.1) - Logging:

If you want to specify new scraping Tasks you go to Data Scraping/logging/Create Logging Data.ipynb" and instantiate a new logging_df for a new user or adjust existing logging_dfs to your wishes. You can adjust the brand and the model of a car that you want to scrape and the earliest and latest initial approval of the cars that should be respected by the scraping. Initial Approval is handled by "start_year" and "end_year". Furthermore, if you want to instantiate a new logging_df and want to add car Brands and models manually, you can adjust the "Data Scraping/logging/Logging_Dictionary.ipynb" which handles with which Brands and models new logging_dfs are instantiated.

1.2) - Webscrape Terminals:

If you dont want to adjust the scraping Tasks or have done it already, you can go to a scraping terminal of your choice, for example "Scraping Terminal Chris" and fill in the required Parameters. Also make sure that a running NordVPN subscription is activated on your engine.

Parameters of the Scraping Terminals:

USER PARAMETERS

user (str): choose an existing user for your scraping task

LOGGING

create_new_logging_df (boolean): Whether you want to instantiate a new logging_df before starting the scraping process. If set to true, this kind of "resets" the scraping tasks for the current user to the values that youre giving in the next parameters.

start_year (int): Only used if create_new_logging_df = True: Set the oldest year of initial approval that you want to respect when scraping.

end_year (int): Only used if create_new_logging_df = True: Set the newest year of initial approval that you want to respect when scraping.

runner(int): Only used if create_new_logging_df = True: Helper variable that is usually set to 0.

user_list (list(of string)): Only used if create_new_logging_df = True: Provide a list of all users that should be respected in the newly created logging_df. Make sure to include your own user here!

SCRAPING LOOP PARAMETERS

n_tries (int): Specify the number of retries when an error occures in the main loop while scraping. Errors typically arise, when AutoScout24.de detects a robot and closes connection. When this happens, the IP-adress is changed and the scraping is attemped additional n_tries-1 times.

sleep_interval (list(of integers)): Interval to pick a random float from to define sleeping time after a request is done on AutoScout24.de. We decided to implement a random number for sleeping to prevent robot detection.

max_pages (int): Define how many pages per search request on AutoScout24.de are searched through. This should generally be set to 20 (maximum amount of pages on AutoCcout24.de).

print_duplicate_url (boolean): Define whether the URL of found duplicates should be printed. This is only set to True for pipeline examination purposes.

RECENCY PARAMETERS

adage (int): Define how old in days the observations searched on AutoScout24.de should be at max. This variable is very useful when deploying a new scraping task, after an old one was completed. In that case, the adage should be set to a low value to prevent to rescrape data that was already scraped with the last scraping task. However, even if duplicates are scraped, they wont be added to the scraped_data, but are dropped immediately. Therefore, this variable just helps to make the loop more efficient.

use_recency (boolean): Whether to scrape only recent data (using adage) or not.

VPN PARAMETERS

area_input (list(of area identifiers for NordVPN): Usually is ['random countries europe 8'] to make sure NordVPN only connects to IP-adresses in Europe.

SAVING PARAMETERS

save_results_local (boolean): Set whether the results df should be loaded and saved to a local path. This is only used for testing purposes, for example when the Cloud Service is not reachable.

do_backup (boolean): If set to true, the pipeline will create a backup csv after each model that is finished. This is a very time intensive step when set to True. Backup solutions have later been implemented within the webscrape_loop_optimized(), which handle that the scraped data is saved (backed up) in any unwanted scenario like loss of internet connection or keyboard-interrupt.

in_out_path (str): Fixed path to local file where scraped data should be stored, when save_results_local is set to True.

in_out_path (str): Relative Path to where the scraped data should be stored on the cloud.

1.3) Data Retrieval:

The scraped data is now saved under "Data Scraping/Scraped Data/user/scraped_data_user.csv". You can find it here or use "Data Scraping/Concat_Scraped_Data.ipynb" to concat all scraped data that is available on the cloud. When using the "Concat_Scraped_Data.ipynb", make sure to provide a list of all users which data should be respected whilst concatting.

Parameters of Concat_Scraped_Data:

user_list (list(of strings)): Define data of which users should be concattenated.

project_location (string): Refine the location of where to find the scraped_data. With our data structure it is set to "", but with different strings the path of the scraped_data can be adjusted if it would lie at another location.

RESULTS:

These steps create the untidy "scraped_data.csv" which is used in the next steps "2) Data Preparation and 3) Data Validation.

2) Data Preparation & 3) Data Validation

LOGIC:

The scripts automatically import the latest version of our scraped_data.csv from “Major Pipeline/Data/Scraped Data.

So in the first step the scraped data is cleaned in a manner that we lose as less as possible observations but keep the data as rich and accurate as possible. This is done using the “Major Pipeline/Data_Cleaning.ipynb” script and its respective functions under “Major Pipeline/Functions/Data_Cleaning_Functions.ipynb”. The results are saved in “Major Pipeline/Data/Cleaned Data/” and are used for Data Analysis and further processing (Dummy Transformation). In this step, also the 3) Data Validation is done to clean the Data from implausible observations, when we implemented the Data_Cleaning_Functions.ipynb by precisely specifying the parameters used in these functions. Furthermore an Outlier Detection is run in this step to eliminate observations which are within 5% of the highest and 5% of the lowest price values in their respective “Manufacturer-Model-Motor” group. The threshold of 5% was found within step 4) Data Analysis.

In the next step, the cleaned data is dummied to make it ready for model computation using the “Major Pipeline/Dummy_Transfromation.ipynb” and its respective functions under “Major Pipeline/Functions/Dummy_Transformation_Functions.ipynb”.

Therefore the results of this step are a cleaned_data.csv and a dummied_data.csv. Please also notice that these steps are finally not done within these scripts but are controlled over “Major Pipeline/Major_Pipeline_Terminal.ipynb” where Data Cleaning, Dummy Transformation and Model Creation is handled in one step, using the parameters specified in the underlying scripts “Major Pipeline/Data_Cleaning.ipynb”, “Major Pipeline/Dummy_Transformation.ipynb” and Major Pipeline/Model_Creation.ipynb”. But we will come to model creation later. Also note, that the fully automated steps that are done in the “Major_Pipeline_Terminal” are logged within the script to check the progress and potential errors.

USAGE:

There is two ways to apply the Data Cleaning and the Dummy Transformation.

1) You can use “Major Pipeline/Major_Pipeline_Terminal.ipynb” to directly run the Data Preparation and Validation according to the parameters set within “Major Pipeline/Data_Cleaning.ipynb” and “Major Pipeline/Dummy_Transformation.ipynb”. Cleaned and Dummied Data will then automatically be saved under “Major Pipeline/Data/Cleaned Data” and “Major Pipeline/Data/Transformed Data/Dummy Transformed”.
2) You can run the Data Cleaning or the Dummy Transformation isolatedly by running the scripts “Major Pipeline/Data Cleaning.ipynb” and “Major Pipeline/Data Transformation.ipynb”. Cleaned or Dummied Data will then automatically be saved under “Major Pipeline/Data/Cleaned Data” or respectively “Major Pipeline/Data/Transformed Data/Dummy Transformed”.

We highly recommend using the fully automated calculation from Way 1) using the Major_Pipeline_Terminal.ipynb. Also, the parameters in the underlying scripts are “optimally” defined and should therefore not be changed. An exception with are the “lower_quantile” and “upper_quantile” used for Outlier Cleaning in “Major Pipeline/Data_Cleaning.ipynb”. These can be adjusted if new data is subject to be less or more noisy or implausible. Therefore in this readme file only these parameters are precisely explained.

Parameters:

lower_quantile (float): Defines how many percent of the observations which are within the lowest price values in their respective “Manufacturer-Model-Motor” group should be dropped.

upper_quantile (float): Defines how many percent of the observations which are within the highest price values in their respective “Manufacturer-Model-Motor” group should be dropped.

Results:

The results of this step are the csv-files “cleaned_data.csv” and “data_dummied.csv” which are saved under “Major Pipeline/Data/Cleaned Data” and “Major Pipeline/Data/Transformed Data/Dummy Transformed” respectively. These will later be used for Data Analysis and for Model Creation.

4) Data Analysis (Christopher)

LOGIC:

Bitte automatisieren und erklären welches „EINZELNE“ Skript ausgeführt werden muss, um die Data Analysis Ergebnisse angezeigt zu bekommen. Das Skript muss in der Major Pipeline liegen und automatisch auf die neusten Cleaned oder Dummied Data zugreifen, um seine Ergebnisse zu erzeugen. Um die neusten Daten zu ziehen, kannst du meine „get_most_recent_file“ Function aus „Major Pipeline/Functions/Basic_Functions“ verwenden. SKRIPTE KOMMENTIEREN (GPT HILFT)

5) Data Visualization (Christopher)

LOGIC:

Bitte automatisieren und erklären welches „EINZELNE“ Skript ausgeführt werden muss, um die Data Analysis Ergebnisse angezeigt zu bekommen. Das Skript muss in der Major Pipeline liegen und automatisch auf die neusten Cleaned oder Dummied Data zugreifen, um seine Ergebnisse zu erzeugen. Um die neusten Daten zu ziehen, kannst du meine „get_most_recent_file“ Function aus „Major Pipeline/Functions/Basic_Functions“ verwenden. SKRIPTE KOMMENTIEREN (GPT HILFT)

6) Presentation of the Result (Janik)

LOGIC:

APP logic erklären und erwähnen, dass in der App unter Expert Mode auch Data Analysis und Data Visualization automatisiert betrieben wird. Außerdem, automatisierte Erstellung von Best Deals, Option Comparison und Car Valuation, und LinReg Modell Creation erwähnen. (Funktionsweise kannst du von mir aus 7) Automation übernehmen und nur ändern, wie die Inputs entstehen). (Funktionsweise von automatisierter LinReg Erstellung muss dir Christopher schicken bzw unten einfüllen und dann kannst du es übernehmen).

7) Automation

The main part of the Automation Process is already talked about in the previous chapters. However, there are some more Front-End related automated processes as well as the automated model creation.

7.1) Automated Model Creation:

Within the Major Pipeline an XGB model is automatically created and is later used for Predictive Tasks (Car Valuation, Best Deals, Option Comparison) by the Front End. This model is created locally and then uploaded to the Application.

A LinReg is automatically created on-the-fly (when user requests a LinReg model in the Expert Mode) and is used for Explanatory Tasks (Expert Mode).

7.1.1) XGB

LOGIC:

The XGB Model is also created within the "Major Pipeline/Major_Pipeline_Terminal.ipynb", where it is automatically created according to the parameters specified within "Major Pipeline/Model_Creation.ipynb". Nevertheless, the XGB-Model can also be created manually within the "Major Pipeline/Model_Creation.ipynb". In both of the scenarios, the trained model will be saved under "Major Pipeline/Models". Furthermore, it automatically uses the latest version of the Dummied Data.

USAGE:

Specify the optimized model parameters under "Major Pipeline/Model_Creation.ipynb" and run either this script or do it within "Major Pipeline/Major_Pipeline_Terminal.ipynb".

Parameters of Model Creation:

model_data (pd.DataFrame): DataFrame containing the data to be used for training and testing.

n_estimators (int): Number of boosting rounds. Optimized is 1000. (This parameter sets the number of trees in the model, more trees can lead to better performance but also can cause overfitting)

max_depth (int): Maximum depth of a tree. Default is 9. (Controls the maximum depth of each tree, deeper trees can model more complex patterns but also increase the risk of overfitting)

learning_rate (float): Step size shrinkage used to prevent overfitting. Optimized is 0.1. (Also known as eta, this parameter scales the contribution of each tree, smaller values make the model more robust to overfitting but require more trees)

early_stopping_rounds (int): Activates early stopping. Validation error needs to decrease at least every <early_stopping_rounds> round(s) to continue training. Optimized is 10. (If the validation error does not improve for a given number of rounds, training is stopped early to prevent overfitting)

data_frac (float): Fraction of data to be used for training and testing. Optimized is 1. (This parameter allows you to use a subset of your data for training/testing, useful for quick experiments)

7.2) Automated Front End Features

7.2.1) Car Valuation via URL

LOGIC:

The car valuation via URL script in “Major Pipeline/Car_Valuation_Via_URL” uses an adjusted form of the scraping mechanic to scrape only a single offer from a user inserted AutoScout24.de URL, which is then send through the cleaning and dummy process to be usable by our trained XGB-Model. The transformed observation is handed over to the model to estimate a fair price for the provided car observation. Furthermore, the average procentual estimation error of the XGB model is used to create a confidence interval for the prediction, such that also the lower_bound and the upper_bound of the fair price can be transmitted. Within the Major Pipeline.

USAGE:

In the WebApp just go to the Car Valuation Tab and click “via URL”, insert URL and press “Predict”. In the Python Environment, go to “Major Pipeline/Car_Valuation_via_URL.ipynb”, adjust the submitted URL parameter and run the script to receive the prediction results for the car observation from the URL.

Parameters for Car Valuation:

url (str): AutoScout24.de URL to the car observation that should be examined

fixed_procentual_error (float): procentual average prediction error from the trained XGB model to calculate kind of an confidence interval for the prediction

7.2.2) Expert Mode (Christopher)

7.2.3) Option Comparison

LOGIC:

The option comparison tab in the Web App or simultaneously from “Major Pipeline/Option_Comparison.ipynb” in the python environment lets the user insert two URLs of AutoScout24.de car offers he found online. The underlying script will then scrape the 2 offers from the provided URLs and run it through the data cleaning and dummy transformation process to make it usable by our XGB model. The user also inserts two further variables. “Planned Usage Years” to indicate how many years he wants to use this car for and “planned kilometers per year” to indicate how many kilometers per year he plans to drive with his new car. The underlying script will then calculate the expected depreciation of the provided car offers over the usage time frame and explicitly return the user how many value depreciation he can expect for both of the cars. This should help the user to come to a buying decision on the used car market by showing him the “real” usage costs and therefore money loss he can expect by using a car, if expected that he afterwards will resell the car.

USAGE:

Go to the option comparison tab in the Web App and fill in the required fields or use “Major Pipeline/Option Comparison.ipynb” in the python environment. In python adjust the parameters

URLS, planned_km_per_year and planned_years and then run the script to get your depreciation results at the end of the script.

Parameters:

urls (list(of strings): Insert 2 URLs of AutoScout24.de car offers which you want to compare.

planned_km_per_year (int): How many kilometers you plan to drive with the cars per year.

planned_years (int): How many years you plan to drive with the car.

Results:

Mileage Depreciation (float €): Value Depreciation due to driven kilometers of the car.

Age Depreciation (float €): Value depreciation due to aging of the car over the time of usage.

Total Depreciation (float €): Total value depreciation over the time of usage.

Offer Price (float €): Offer Price of the car.

7.2.4) Best Deals

LOGIC:

The Best Deals Feature from the Web App or simultaneously from “Major Pipeline/Best_Offers.ipynb” in the python environment lets the user specify a car model he is interested in, over the manufacturer and the car model variable, as well as the maximum respected kilometers on a car and the latest initial approved allowed for his subselection. The script will then run through the dummied data and subselect it as specified. Then it will make predictions on the fair prices of the observations and calculate the difference between the offer price and the fair price. It will then order the resulting dataframe the price difference, to show observations first, which have a higher fair price than offer price (because that indicates that the offer price is rather low). In the next step, an adapted scraping mechanism is used to run through the sorted data frame and check if the cars are still available, until it found the 10 most lucrative still available offers.

USAGE:

With the WebAPP go to “Best Deals” and fill in the requested fields. In the python environment go to “Major Pipeline/Best_Deals.ipynb”, fill in brand, manufacturer, max_mileage and initial_approval. Make sure you have a running NordVPN subscription installed and run the script. The rest of the parameters should be kept as they are.

Parameters:

manufacturer (str): Choose the brand of the car model youre interested in.

car_model (str): Choose the exact car model youre interested in.

max_mileage (int): Choose the maximum allowed mileage for the cars youre interested in.

initial_approval (int): Choose the earliest allowed initial approval for the car youre interested in.

n_days (int): Maximum of how many days back the considered data has been scraped. Is used to make this function more efficient, and not to respect to old offers from the scraped data, as it is probable that they are not available online anymore.

# SCRAPING PARAMETERS

change_vpn (boolean): Whether the VPN should be initialized and changed or not.

area_input (string): Usually ['random countries europe 8']. Input parameter for nord_vpn switcher, which declares from which pool of IPs NordVPN can pick one when changing the IP-adress.

sleep_time (float): Sleeping time between url requests in seconds.

# BEST OFFER PARAMETERS

n_best_offers (int): Declare how many best offers should be shown.

7.2.5) Developer Tab

LOGIC:

Hidden and password-secured developer tab for the project developers. Supposed to handle the whole Major Pipeline to automate Data Cleaning, Dummy Transformation and XGB Model Creation directly within the WebApp. But was not finally implemented yet. Therefore you can just upload a cleaned Dataset here, using the Password “1234”, which will be used for the Expert Mode.

8) Version Control and Collaboration (Christopher)

LOGIC: