MIE 451 Decision Support Systems
Assignment 2: Machine Learning (ML)

This assignment allows students to apply machine learning knowledge through sales data regression using supervised learning techniques and to perform performance analysis of the learned regressor. In addition, this assignment contains a competitive component to encourage students to explore feature sets, feature representations and hyperparameter tuning.

- Programming language: Python (Google Colab Environment)
- Due Date: Posted in Syllabus

Marking scheme and requirements: Full marks will be given for (1) working, readable, reasonably efficient, documented code that achieves the assignment goals, (2) for providing appropriate answers to the questions in Jupyter notebooks (`ml_assignment.ipynb` and `ml_assignment_Q4.ipynb`) committed to the student's github assignment repository, and (3) attendance at the code review session, running your solution notebook for instructors, and providing clear and succinct answers in response to instructor questions regarding your solution.

Please note the plagiarism policy in the syllabus. If you borrow or modify any *multiline* snippets of code from the web, you are required to cite the URL in a comment above the code that you used. You do not need to cite tutorials or reference content that demonstrate how to use packages – you should certainly be making use of such content.

What/how to submit your work:

1. All your code should be included in the notebooks named `ml_assignment.ipynb` and `ml_assignment_Q4.ipynb` provided in the cloned assignment repository.
2. Commit and push your work to your GitHub repository in order to submit it. Your last commit and push before the assignment deadline will be considered to be your submission. You can check your repository online to make sure that all required files have actually been committed and pushed to your repository.
3. A link to create a personal repository for this assignment is posted on Quercus.

Notes that you should pay attention to

1. The same auto-grade restrictions apply for this assignment as they did for the previous assignments. Please check the pinned posts about autograder on Piazza for more information.
2. The autograder will be making 2 separate commits. The first will run test cases for Q1-Q4. The second is for the competitive portion of Q4 where we'll test performance on our own train and test sets. The second commit may occur hours after the first. Please do not push any changes to your repository in between these two commits, or we will consider your submission late.
3. DO NOT use global variables in your functions. You can only use variables passed into functions.
4. DO NOT change the names of functions.
5. During the code review, we will ask questions about your free text answers and your results in general as well as some details about your code and possible alternative approaches you might have taken. If you made both an on-time and late submission, you will have to tell us which version we should use for code review with a 30% deduction applying to the code review if you wish us to use your late submission. Note: you will receive a zero for the code review if you did not make an on-time or late submission – in this case you are considered to not have submitted your assignment.

This assignment has *7 points* in total, allocated as follows:

- Auto-grading points(5 points):
  - Q1: 1 point
  - Q2: 1.5 point
  - Q3: 1.5 point
  - Q4: 1 point
- Code review (2 points)

Points will be deducted from the auto-grading points for missing or incomplete text answers.

1 Before the Introductory lab

In the lab you'll familiarize yourself with several python machine learning libraries;

- **scikit-learn** a powerful machine learning library for python.
- **pandas** a powerful python data analysis toolkit
- **numpy** and **scipy** powerful computing packages for python
- **matplotlib** a python 2D plotting library

The scikit-learn documentation can be found on its website at `http://scikit-learn.org/stable/`

The pandas documentation can be found on its website at `http://pandas.pydata.org/pandas-docs/stable/`

The numpy and scipy documentations can be found on its website at `http://docs.scipy.org/doc/`

The matplotlib documentation can be found on its website at `http://matplotlib.org/`

You just need to execute the import statements at the top of the lab ipynb notebook to import these standard Python libraries.

2 In the Introductory lab

The lab can be found in your assignment-ml repository under the file `lab_ml.ipynb`. Clone the repository and open the Jupyter Notebook in Google Colab to follow along. In this introductory lab, we will guide you through loading the dataset titled `rossmann_store_sales_lab` and running two machine learning models: Linear Regression and Gradient Boosting Regression. This dataset is sourced from a 2015 Kaggle competition `https://www.kaggle.com/competitions/rossmann-store-sales/data)`, hosted by Rossmann, one of the largest drugstore chains in Europe. The goal is to predict daily store sales based on a set of observed features, including promotions, competition, school and state holidays, seasonality, and locality.

After completing the lab, you should be familiar with:

- **Exploratory Data Analysis**: Using histograms, bar plots, scatter plots, and mutual information to understand the basic structure of the dataset. You will learn how these features can be used to inform the model-building process.
- **Feature Engineering**: Identifying missing values and imputing them based on domain knowledge. You will also learn to derive new features from existing ones to improve model performance, and explore how adding external datasets can enhance prediction accuracy.
- **Linear Regression**: Fitting a linear regression model using the `scikit-learn` package, interpreting coefficients in the context of the features, and understanding learning curves. We will cover concepts such as overfitting and underfitting, as well as Ridge and Lasso regularization to mitigate overfitting issues.

- **Model Evaluation**: Understanding the importance of validation datasets and cross-validation. You will learn to plot learning curves and evaluate your final model using a new test dataset.
- **Gradient Boosting**: Implementing gradient boosting using both `scikit-learn` and `XGBoost`. We will also discuss feature importance and how to tune hyperparameters to improve model performance.

3 Main Assignment

In the introductory lab, you worked with historical sales data from 200 Rossmann stores and predicted sales using eight observed features. For this assignment, you will work with a larger dataset containing 800 stores and more features. The final goal remains the same: to predict store sales of some date using given features.

From Questions 1 to 4, you will explore different techniques to train your model. For Question 5, you will compete with your classmates to achieve the best Root Mean Squared Error (RMSE) in sales prediction.

We provide a template notebook for the coding portion of the assignment, which must be used for your submission. Autograding requires that all function names remain unchanged and that the notebook executes in a single pass in the Colab environment (please verify this before submitting). You are expected to complete the missing parts of the functions in the template notebook and produce experimental results. Each function you need to implement is marked in the assignment questions.

You will be provided with two datasets, `train_data.csv` and `test_data.csv`. `train_data.csv` contains essential information to train your model, with the following observed features:

- **Store** - a unique ID for each store.
- **Date** - the date of each record.
- **DayOfWeek** - the day of week of given date, such as 1 = Monday, 2 = Tuesday, ...
- **Customers** - the number of customers on a given day.
- **StateHoliday** - indicates a state holiday `a` = public holiday, `b` = Easter holiday, `c` = Christmas, `0` = None.
- **SchoolHoliday** - indicates if the (Store, Date) was affected by the closure of public schools.
- **StoreType** - differentiates between 4 different store types: `a`, `b`, `c`, `d`.
- **Assortment** - describes an assortment level: `a` = basic, `b` = extra, `c` = extended.
- **CompetitionDistance** - distance in meters to the nearest competitor store.
- **Promo** - indicates whether a store is running a promotion on that day.
- **Promo2** - Promo2 is a continuing and consecutive promotion for some stores: `0` = store is not participating, `1` = store is participating.
- **PromoInterval** - describes the consecutive intervals Promo2 is started. `4` means there are four consecutive months in any given year for that store.

- **State** - (Optional) The geographical states (or province) that each store was located in, which can be used to match with weather data, Google trends, and other external datasets mentioned in the lab.

The value in the column `Sales` is the turnover for any given day, which is the target variable you are predicting. The `test_data.csv` dataset contains a list of records with the same features, which you will use to evaluate your model's performance. **You should only use the `test_data.csv` after you have a satisfactory model to accurately evaluate its performance.**

Please note the following:

- Do NOT change the function names, and any provided code, we need those for grading.
- Do NOT use/refer to global variables in the functions. This will cause your code to fail during autograding because we execute functions independent of any content outside the function.
- If you need to use **external libraries** that are not provided in the lab, please ask for approval on Piazza.

Please answer the following questions:

Q1. Exploratory Data Analysis

(a) Before building and training a model, it's important to first understand the information available in the dataset. Please complete the `basic_stats(data)` function in the provided template notebook to extract and fill in the following variables with the necessary information using the input DataFrame `data`:

- **num_of_total_records**: The total number of historical sales records available in `data`
- **num_of_features**: The total number of features (including 'Sales') available in `data`
- **num_of_total_unique_dates**: The total number of unique dates available in `data`.
- **num_of_unique_stores**: The total number of unique stores available in `data`.
- **mean_sales**: The average daily sale among all records in `data`.

In the markdown cell, please list three additional pieces of information related to the dataset, different from those listed above, that you, as a data scientist, would want to know to help design your model. [1 sentence description for each piece of information].

(b) Complete the following functions to visualize the training dataset using the Seaborn library:

- `schoolholiday_count(data)`: Create a count plot that visualizes the number of records affected by school holidays (`SchoolHoliday = 1`) versus those not affected (`SchoolHoliday = 0`).
- `sales_vs_promo2_bar(data)`: Create a bar plot to compare the average sales of stores during dates without consecutive promotions (`Promo2 = 0`) and during dates with consecutive promotions (`Promo2 = 1`).
- `sales_vs_competition_scatter(data)`: Create a scatter plot where the x-axis represents `CompetitionDistance` for each record, and the y-axis represents the corresponding sales value.

All three plots should have informative titles and axis labels. Each function should return the `pyplot.axes` from the plot you created (Hint: Please refer to the lab for an example). You can execute the provided code to display your plots.

For each plot, write 1-3 sentences on any insights you observed regarding the features. If no insight is found, provide a possible reason that might explain the lack of insight.

(c) It's always a good practice to check whether the provided dataset contains any missing values. Complete the `missing_data(data)` function to check for missing values in each feature of `data`. The output of this function should be a dictionary where the keys are the feature names (as shown in the columns) and the values are the counts of missing values for each feature.

In the provided markdown cell, if there are features with missing values, comment on which features have missing values and suggest one method to impute the missing values.

Q2. Features Engineering

In this question, you will clean the data by converting the format of given features, evaluating the mutual information between observed features and our target feature `Sales`, and deriving new features to help sales prediction.

(a) Complete the `clean_data(data)` function, which takes `data`, a DataFrame (i.e., the `train_data.csv`), as input and outputs the cleaned data according to the following instructions:

1. Convert the `Date` column into three columns named `Year`, `Month`, and `Day`. For example, `2015-07-31` should be converted into `2015`, `7`, `31`.
2. Create three indicator variables for the `Assortment` column, with names `Assortment_a`, `Assortment_b`, and `Assortment_c`. For example, if `Assortment` is `'a'`, then `Assortment_a` should be 1 and all other indicator variables should be 0.
3. Drop the `Date` and `Assortment` columns from the dataset.

In the provided markdown cell, suggest another feature that you think converting into a different format would help improve the model training. Alternatively, you may suggest a different approach for converting the features listed above that could potentially work better than the given one. For either option, explain the reason in 1-3 sentences.

To ensure everyone has the same dataset for the rest of the questions, we have provided a cleaned training dataset in the variable `cleaned_train_data`. Beyond the steps outlined in the previous questions, this dataset has been processed as follows:

- Created four indicator variables (one hot encoding) for the `StateHoliday` column, with names `StateHoliday_a`, `StateHoliday_b`, `StateHoliday_c`, and `StateHoliday_0`. For example, if `StateHoliday` is `a`, then `StateHoliday_a` should be 1 and all other indicator variables should be 0.
- Created four indicator variables for the `StoreType` column, with names `StoreType_a`, `StoreType_b`, `StoreType_c`, and `StoreType_d`. For example, if `StoreType` is `a`, then `StoreType_a` should be 1 and all other indicator variables should be 0. Dropped `StoreType_a` to avoid multicollinearity.

- Converting `Months` and `DayofWeek` into indicator variables and renamed them accordingly. Dropped January and Monday to avoid multicollinearity.
- Dropped feature `State`
- Note that the feature `store` is not converted into indicator variables due to there being more than 1000 stores in this dataset (**this is not recommended in general practice (see lab for more detail)**). You can choose to convert it as a new feature later in Q2d.

Please use the `cleaned_train_data` for the rest of the questions.

(b) Complete the `calculate_mi(data)` function to calculate the Mutual Information (MI) between the target variable `Sales` and all features (including `Sales` itself) within `data`. This function should return a Pandas Series, where the index is the feature name, and the value represents the mutual information between `Sales` and the corresponding feature. The Series should be sorted from features with the highest MI to the lowest. (Hint: please refer to the lab notebook)

(c) In the provided markdown cell, please answer the following questions:

- What does mutual information tell you about the dataset? How could MI help with sale prediction using machine learning? [1-3 sentences]
- Which feature has the highest MI (except 'Sales' itself), and which feature has the lowest MI? [1-3 sentences]
- Please state one potential problem when calculating MI within a dataset that has categorical features.
- Do you think this result makes sense? If yes, select a feature and explain why this feature would have high (or low) MI using your background knowledge. If no, suggest a feature that conflicts with your domain knowledge. [1-3 sentences]

(d) Now, let's derive at least three additional features from the existing features using the cleaned training dataset that you believe will enhance daily sales prediction in Rossmann stores.

Please complete the `new_features(data)` function, which takes a DataFrame `data` as input (specifically the cleaned training dataset `cleaned_train_data`). You should ADD at least three new columns to `data`, where each column contains one of your newly derived features for the corresponding sample, and return the processed DataFrame.

DO NOT derive any features using the target feature 'Sales'. When the information in target features is included in the features used for training, it can lead to overly optimistic performance metrics during training. This is because the model has access to information that would not be available at test time, where the target variable is unobserved.

The provided code will retrieve the DataFrame returned by the `new_features(data)` function and calculate the Mutual Information (MI) using the function you implemented in Q4b. Based on MI, do the features you created rank higher or lower in the MI ranking? Select one feature you created that ranked high in MI and explain your reason for creating this feature. If none ranked high, select one of them and explain why it may not have ranked well.

Q3. Linear Regression

(a) First, let's fit a linear regression model using the original features available in the dataset. For this part of the question, we have split the training and validation datasets for you. Complete the `Q3a(X, y)` function, which takes the observed features from the training data as a DataFrame (`X`), the target feature `Sales` as a Series (`y`), and the value of the hyperparameter `alpha`. Using the input training data, train and return a Linear Regression model with Ridge regularization from `sciki-learn`. The hyperparameter `alpha` should be 0.5 for this section.

The provided code will generate a DataFrame that contains the coefficients for each observed feature using the linear regression model you created. What do these coefficients tell you about the features and the model?

(b) Now, let's tune the hyperparameter `alpha` in your Ridge model. In the `Q3b(X_train, X_train, X_val, y_val)` function, specify five suitable values of `alpha` for this dataset in the `hyper_params` list.

Next, train a Ridge model with `X_train` and `y_train`. Please use one of the five values you listed `alpha`. Predict the daily sales using the observed features in the validation dataset `X_val` and calculate the Root Mean Squared Error (RMSE) using the ground truth `y_val`. Repeat this process for all five values of `alpha`.

Store the validation RMSE in a dictionary, where the key corresponds to the `alpha` value used in the model. For example, if you are tuning `alpha` with `hyper_params = [0.1, 0.2, 0.3, 0.4]`, the resulting dictionary should be structured as follows:

{ 0.1: mean RMSE of the model with `alpha = 0.1`,
0.2: mean RMSE of the model with `alpha = 0.2`,
0.3: mean RMSE of the model with `alpha = 0.3`,
0.4: mean RMSE of the model with `alpha = 0.4` }

Create a line plot using the Seaborn library to visualize the relationship between RMSE and `alpha`. The y-axis should represent the RMSE of the validation dataset, while the x-axis should show the corresponding `alpha` values. Be sure to include a proper title and axis labels in the plot. `Q3b(X_train, X_train, X_val, y_val)` should return the RMSE dictionary.

Finally, the provided code retrieves the dictionary returned by `Q3b(X_train, X_train, X_val, y_val)` and generates a `DataFrame` sorted by the RMSE. Report the value of your tuned `alpha` that yields the lowest RMSE.

(c) Now, let's try the features you created in the previous Q2(d). Instead of using a single validation dataset, a better practice to evaluate a model is by using cross-validation. In the `Q3c()` function, which takes the completed training dataset with observed features (including the ones you created) (`X`) and the target feature `Sales` (`y`), you can use all features, or select a subset of features that you think are important for model training (but must include the features you created). Train a Ridge model using 5-fold cross-validation and calculate the Mean RMSE across all folds.

Compared to the linear model with original features, does the RMSE improve? If not, explain a possible reason that might affect the performance of the model.

(d) Now, let's explore feature selection based on the Mutual Information (MI) result from the previous Q2b question. In the `Q3d(X,y)` function, which takes the completed training dataset with observed features as `X` and the target feature `Sales` as `y`, you should train a Ridge model

using the top K variables with the highest MI, based on the results from the previous Q2b question. Evaluate the model's performance by calculating the mean RMSE using 3-fold cross-validation.

Within `Q3d(X,y)`, create a line plot using the Seaborn library to visualize the mean RMSE against the number of top K variables used in model training, where K should range from 1 to 20. Be sure to include a proper title and axis labels in the plot. Please also include the 95% Confidence Interval using your 3-fold validation RMSE, shaded within the plot (Hint: please refer to the lab material). `Q3d(X,y)` should return the `plt.Axes` of this plot. (For this question, you can choose a value of `alpha` based on previous tuning. But for general practice, you should tune `alpha` along with K)

(Note: You should not use any global variables; the names of the top 20 features with the highest MI score should be manually specified within this function.)

Comparing the validation results from the previous two parts (using original features after tuning and all features) with the lowest RMSE in the plot, discuss whether the RMSE has improved [1-3 sentences]. If it has not, provide a possible explanation for factors that may have influenced the model's performance [1-3 sentences].

(e) Now, let's evaluate the best model you created so far with the testing dataset. Select the best model across all previous questions.

The `Q3e(train_data, test_data)` function takes a cleaned training dataset and a cleaned testing dataset as input. Your job is to retrieve the target feature (`y_test`) and observed features (`X_test`) from the testing dataset. Then, create the model that performed the best based on previous questions, train the model using the training dataset, and evaluate the performance of the model using the testing dataset with RMSE. The function should return the RMSE of the testing dataset.

Compared to the cross-validation results, does the best model perform better or worse on the test dataset? Please provide a possible reason for this difference (1-3 sentences).

Q4. Sales Prediction Competition

This question is the competitive portion of the assignment and should be answered in the `ml_assignment_Q4.ipynb` notebook. In this question, you will again build a regression model to predict the daily sale price of the Rossmann Store. However, there is no limitation on the features you use. You can use any model (**within the Sklearn and Xgboost package**) that you think has the best performance. You will be putting together a model to optimize the **Root Mean Squared Error**. You should clean the data, train your best model, and predict the sale of the test dataset in the provided functions `Q4` (see below for more details). There is a runtime limit of **4 minutes** for this question (running the `prediction(train_data, test_data)` function). Models that exceed this limit will receive a 0 mark for this question. (Hint: If you choose to use Gradient Boosting Tree, the implementation in Xgboost is substantially faster compared to implementations in the scikit-learn package.)

(a) Implement the function `data_cleaning(raw_data)` that selects any subset of features, and applies any feature engineering techniques to preprocess the `raw_data` DataFrame for model training. This function should return a cleaned dataset, containing both observed and target features (Sales). Please make sure the following requirements are met:

- **Do not** modify the column name `Sales`.

- Ensure that the order of the rows in the returned dataset retains the same order as the input dataset. (i.e., Do not change the order of samples in the dataset.)
- `data_cleaning(raw_data)` should also handle the raw test dataset (e.g., `test_data_assignment`) as input, where the `Sales` column is not present in the dataset. Therefore, **DO NOT derive any features using the target feature 'Sales'.** When the input is a test dataset, the returned dataset will only contain the observed features.
- You may use the provided `weather_data.csv` in `sale_data` folder, which contains wether data in different regions. The column `State` can match with the feature `State` in the train/test dataset. (You can assume that all possible `States` are available in the given training dataset)

(b) Implement the `model_training(train_data)` function takes cleaned training data as input and trains the **best** machine learning model of your choice. This function should return the trained model that's ready to be used for test dataset prediction.

(c) Implement the `prediction(train_data, test_data)` function that:

- Using the `data_cleaning()` function to preprocess the training (`train_data`) and testing (`test_data`) dataset.
- Using the `model_training()` function with the cleaned training dataset to retrieve a trained model for sale prediction.
- Predicting the daily sales of samples in the testing dataset.

The function `prediction(train_data, test_data)` should return a `DataFrame` with a single column containing the predicted daily sales for the testing dataset. The AutoGrader will use this prediction to evaluate the RMSE of your model (please see below for grading details).

You can use the `test_data.csv` and the provided code to validate your function and see an example of the code AutoGrader used in the evaluation. However, for your final score in Q4, we will use a different set of test cases.

Please avoid using GPU (e.g., CUDA) in your solution, as the autograder does not have a dedicated GPU to support this task.

(d) In markdown cells, provide:

- A clear and concise description of your chosen feature set and any additional features you used through feature engineering or external sources.
- The name of the regression model you chose
- Why you choose the feature set and model?

Your score for Q4 will be calculated using the min-max normalization shown below:

$$Q4\_score(RMSE_{yours}) = \max(0, \frac{500 - RMSE_{yours}}{500 - RMSE_{max}}) \quad (1)$$

Where $RMSE_{yours}$ is your RMSE tested using hidden test cases and $RMSE_{max}$ is the highest RMSE score obtained across all student submissions. You will not receive the credit for Q4 if your RMSE score is higher than 500 and you will receive the full credit(1 point) for Q4

if your score the highest among all students.