CSIS 3290 Fundamental of Machine Learning
Mini Project 01

Due date: Oct 23 17:00

NOTE:

Instructor may ask the project group or individual members questions about the actual work and contribution after submission. The questions may be very specific (E.g., the lines of code written by a member and the explanation of the codes.) It may be done via email or in-person after the class without advanced notice. Failing to answer the questions satisfactorily will result in mark deduction. Please ensure all group members have roughly even contribution in the project, especially the coding part, and have full understanding on the whole project.

Grouping:

You need to form groups of 3 to complete the project in this course. Sign your project group list on the Wiki page in Blackboard.

One of the members in your group, the group leader/captain, is responsible for submitting the project. Marks will be deducted if more than one member of your group submit the same/different project.

Project Description:

In this project you will be working on an unclean dataset that contains information of several thousands cars' resale value that were obtained from UK's craiglist. You will need to implement several linear regression techniques to predict the resale value of those car. There are two parts in this project.

Project Submission Requirements

Project01.ipynb (or a zip file if you have any extra files): In this Jupyter notebook, you have to utilize different cells (code/markdown) to clearly indicate and explain every step. Your Jupyter notebook should include all the markdown texts signifying the steps with correct heading, python code, comments/analysis, and visualizations as stated in the following instruction. Note: You need to create the appropriate markdown headings for each section mentioned below. Codes should have some short comment describing the statement. Adding a markdown cell containing text before specific actions performed is appreciated.
(*Note: Restart the kernel and re-run all cells of the notebook before submission. Substantial marks will be deducted for cell errors.*)

Part A. Regression Modelling Requirement

You are required to create several linear regression models to predict the pricing of a certain used cars make. In order to complete all the modeling requirements:

- You need to use all the knowledge in Python coding and/or data wrangling techniques covered in the class
- You may need to perform some research to implement some models that have not been covered in the class

1. **Title, Name and References**
   Include a title of your regression project. Include your name and student ID. Add information about any references you used to help complete the project

2. **Library import and data loading**
   Import all the required important library and load the provided unclean dataset, i.e., **unclean_data.csv**.

3. **Data Analysis, Preparation and Wrangling**
   Before you start, please take a look at the original csv file to find out about the data. Then, explore the dataset in your Jupyter notebook. Have a peek of the data. You should notice that the dataset is not clean. The followings are the issues that appear in the dataset:
   a. There are missing rows (rows that have all NaN)
   b. There are missing values in some columns
   c. There are fields that use a wrong datatype, i.e., year is in float64 instead of int64. Hint: You can use df.dtypes to look at the datatypes of each columns
   d. There are columns that have mixed values, i.e., string and numbers
   e. There is a column with currency symbol
   f. Some numerical values are displayed with comma format
   g. Some columns are using different standard values, i.e., engine size and engine size2
   h. Some columns do not have valid names
   i. There is a column with categorical data, i.e., transmission and fuel type
   j. There are some unneeded columns

   The followings are actions you need to perform to handle the abovementioned issues.

If you are not able to perform the actions as required below, instead of spending too much time working on the data preparation without completing any regression models, you can load the cleaned data. You will not get any mark for the data preparation, but at least you may get mark for some of the regression modelling.
Note: the provided cleaned dataset does not have exactly the same data as the one produced from the following data preparation steps.

Handling missing rows

You should delete the rows where all values are missing. After you drop those rows, you may want to re-index the dataframe after that. Hint: you can use `df.reset_index(drop=True)` for this purpose. The parameter drop=True is needed such that the new index will not be added as a new column. Remember that many dataframe functions are performed on the copy of the dataframe. You need to make sure that the dataframe that you will work on is the one that has its index changed.

Handling the currency symbols and comma format

The values in the price column is stored as string. We need to replace the string currency symbols with empty space. Similarly, you should change the comma character with empty space. Hint: you can use ***lambda function*** and `str.replace()` for this purpose.

Note: If you chain both lambda in a single statement, you may get a Python warning. You can ignore the warning. Or, you can modify the statement such that you will perform the action on a copy of the dataframe (or the corresponding series), and assign it to a variable. You can then assign this variable to replace the price column. You might get another warning when you do that. There is another way to avoid the warning, but you can just ignore it for now.

Handling non-valid column names

You should rename the following columns: “fuel type”, “fuel type2”, “engine size”, and “engine size2” into “fuel_type”, “fuel_type2”, “engine_size” and “engine_size2” respectively. A short way to do that is by using `df.rename(columns=dict, inplace=True)`. For example, to change a column name from “foo” to “bar, you should use `df.rename(columns={'foo' : 'bar'}, inplace=True)`

Handling missing values in columns: fuel_type

You should notice that there are missing values in fuel_type and fuel_type2. You should see that it is easier to fill the missing values in fuel_types2 column with values from column fuel_type than the opposite. In order to do that you can use the df.fillna(values), i.e., `df.fuel_type2.fillna(df.fuel_type)`. Remember that this action is performed on the copy of the dataframe. Make sure to replace the fuel_type2 column with the above statement. Note: Since fuel_type2 is more complete, please drop fuel_type column. After that, rename the fuel_type2 column into fuel_type.

Handling missing values and comma format in columns: mileage

You need to perform similar action for the mileage. However, the order should be reversed since column mileage has more complete data than mileage2. In addition to that you should also then remove the comma format using str.replace() in the mileage column. Note, you may need to parse the numerical value of mileage to string in order to remove the comma. Use `pd.to_numeric(df['mileage'],errors='coerce')` to convert mileage into numeric. After that, you can drop the mileage2 column.

Handling missing values and different value standard in columns: engine_size

The engine_size2 column is more complete data than the engine_size one. You should fill the missing values in engine_size2 with values from the engine_size column. After that, you should also notice that some engine values are in liter while others are in cc. Some values have one decimal while others have more than three. We want to make sure that the values are using the same standard. So, we need to divide the values that are greater than 1000 by 1000 and round it into one decimal. But this only be performed on numerical values, which is not the case for the engine_size2 column. Use `pd.to_numeric(df['engine_size2'],errors='coerce')` to convert engine_size2 into numeric. Then use a lambda function to standardize its value. An example lambda function you can use is `lambda x: round(x/1000,1) if x>1000 else round(x,1)`

After you are done with the above step, you can drop column engine_size. Next, you should rename engine_size2 into engine_size.

Handling wrong data type: year

The above steps will remove almost all the NaN values in the dataset. You can then call dropna() to remove any remaining NaN. After that, you can change the data type of the year column into integer format with the following statement `df['year'].astype('int64')`. Note, if you did not perform the dropna() beforehand, you will get an error message.

Handling unneeded columns

You can drop the model and reference columns.

Handling categorical data

The transmission and fuel_type have categorical values. In order to use them in our model, we need to change the categorical data into *dummy values* that indicates the absence or presence of a feature with a 0 or 1. For example, instead of a single transmission column, we can have Automatic, Manual, Semi-Auto and Other columns; each having value either 0 or 1.

Use the following statement: `pd.get_dummies(df['transmission'])` to get the dummy values. In order to avoid confusion, you should save the statement into a variable. After that, you can add the new columns into the dataframe using df.join() statement. For example, assuming that the variable used is transmission, you should write `df.join(transmission)`. Remember that functions in dataframe work on a copy of itself if you don't use argument inplace.

Perform the similar action to the fuel_type column. Before you join the dummy values, you should notice that one of the dummy values columns is named as 'Other' that is use in the transmission's dummy values. You should rename the fuel type dummy values of Other into something else before joining them to the dataframe.

Drop both transmission and fuel_type columns.

Saving the cleaned data into csv

Make sure to change the format of mileage and price column into numeric using pd.to_numeric() if you have not done so. After finishing all the above steps, your data is well prepared to be used for

analysis. You should have about 3900 rows of data now with 12 features. Save the cleaned data using to_csv() function as *cleaned_data.csv*.

4. **Exploratory Data Analysis and Visualization**
   You should now perform your EDA and visualization on the prepared data as we have done in the class demo. Additional actions and visualizations performed in this part can give you any additional weight in your mark.

5. **Feature Observation and Hypothesis**
   Provide analysis on the features available, and your hypothesis for the regression model. Feature observation from EDA and visualization that leads to feature transformation for the requirements number 8 below should also be mentioned here.

6. **A Simple Linear Regression Model**
   Create a simple regression model to predict the used cars pricing. You can use one or a few features in this linear regression. You should justify why you select those features in your model. Please refer to the lecture demo in creating the linear model. You should use 90:10 for training and testing. You should also evaluate the model by calculating the RMSE and R^2 metrics, plot the predicted vs actual and print the model coefficients.

7. **Linear Regression Model with Lasso/Ridge**
   You can choose to use either Lasso or Ridge for this model. You should use all the available feature in this model and decides on the alpha value for either the Lasso or Ridge model. You should use 90:10 for training and testing. The splitting needs to be the same as Step 6. You should also evaluate the model by calculating the RMSE and R^2 metrics, plot the predicted vs actual and print the model coefficients for the alpha that produces the best model.

8. **Polynomial Regression Model (with Lasso/Ridge)**
   Based on your observation of some of the features, you should propose polynomial regression model to better predict the car pricing. For this model, you need to import PolynomialFeatures from the sklearn.preprocessing and decide on the degree of polynomial to be used in your model. You can create a simple polynomial regression model or polynomial regression model with Lasso or Ridge. Note: in order to complete this part, you should perform your own research on how to create a polynomial degree of the features. You should use 90:10 for training and testing. The splitting needs to be the same as Step 6. You should also evaluate the model by calculating the RMSE and R^2 metrics, plot the predicted vs actual.

Part B. Competition

In this part, your group competes with other groups in performance.

1. **Dataset**
   Use the competition_data.csv, which is the cleaned dataset prepared by Instructor based on Part A. Load this dataset into your Jupyter notebook. You MUST NOT remove/add any records from/to the dataset.

2. **Training/Test Split**
   Your group must use 90% training and 10% testing using the random seed 1024 to split the data. (e.g.,: X_train, X_test, Y_train, Y_test = train_test_split(features, response, test_size=0.1, random_state = 1024))

3. **Model Creation**
   Create regression model (Linear/Polynomial/Simple/Lasso/Ridge) using training set. You may use all columns, a subset of the columns, derived columns; you may use the original or transformed values of any columns when creating the model.

4. **Evaluation**
   Evaluate the performance of the model in the test set. The metrics are RMSE and R^2 metrics. The ranking of the competition is determined by the RMSE in test set.

C. Member Contribution

In addition to the proposal, each group needs to submit a peer evaluation matrix. Each cell should be a number between 1 and 4, which reflects how a member thinks the contribution by another member. The evaluation is opened to open to all members of your group (i.e., Every one can see how others grade on you), so that each member knows how to enhance their contribution in the project.

(Hint: You may refer to this link to see how to create a table in a Jupyter Markdown cell.)

Evaluator \ Evaluatee	Member 1	Member 2	Member 3
Member 1			
Member 2			
Member 3			

Here is the rubric on how to evaluate your team members:

1 Point: No or very little contribution to the project; cannot deliver artifacts or largely miss the agreed deadline; showing no or very little passion in development.

2 Points: Little contribution to the project with no negative effect to the group; sometimes cannot deliver artifacts or miss the agreed deadline; mainly follow other members' idea and instructions.

3 Points: Fairly large and positive contribution to the project; can handle most of the assigned tasks and deliver artifacts on time;

4 Points: Large and positive contribution to the project; can help members to tackle problems; pro-active and passionate in the development.

D. Project Grading Criteria

The project will be graded on a scale of 20 points.

Criteria		**Grading**
Project submitted, named properly with all files included in their corresponding folders to Blackboard.		1
Part	**Detail**	
Jupyter	The data preparation steps from the uncleaned data were performed completely. The csv file of the cleaned data is included in the submission.	3
	The EDA was performed perfectly along with the plots and their analysis	3
	Feature observation and hypothesis were adequately provided	1
	Simple Linear Regression was completed perfectly	3
	Linear regression with Lasso/Ridge was completed perfectly	3
	Polynomial Regression (with Lasso/Ridge) was completed perfectly	3
Competition	One best group: 3 Other groups with reasonable effort: 2 Others groups with simple approach: 1 No submission/problematic implementation/invalid approach: 0	3