MINI PROJECT 1 Spring 2020 Machine Learning End-to-end project, predicting vehicles MPG.

Notebook 80% (code and documentation)
Originality and creativity 20%

Participants name	Code Section	Report Section	Documentation Sec	Presentation Sec
Nathan Rabideau	X	X	X	X
Josh Baldwin	X	X	X	X
Steve Zhou	X	X	X	X

Submission deadline: Feb 6th at 11:59 pm.
Presentation deadline: Feb 6th at class time.

Deliveries:

- A Notebook file which include code and descriptions

Note: Read chapter 2, (End-to-end project) and let's apply this methodology but using a different dataset. In this case, we will use the Auto MPG dataset.

BONUS: 10 Points in the first exam for the group showing the less minimum error in the performance evaluation of the model, 8 for second place, and 5 for third place. You should make and analysis of the steps that help you to minimize the error, possible factors include the right choose of parameters, the way to encode the features, the type or lack of feature scaling, etc.

Dataset: Detail information of the dataset can be found here (https://archive.ics.uci.edu/ml/datasets/Auto+MPG) . The dataset has 398 instances, 8 attributes, and the target variable (MPG).

Resources: Chapter 2 End-to-end projects is the main resource for this project. Some of the function could be out of date. However, and updated version of the code could be found in the GitHub account of the book. Download the code of the textbook from https://github.com/ageron/handson-ml and load the code for chapter 2, namely (handson-ml/02_end_to_end_machine_learning_project.ipynb)

Project description: Your job consists in predicting the target variable (MPG) by using all or some of the most relevant features of the dataset.

Steps:

1. Get the data.
2. Discover and visualize the data to gain insights.
3. Prepare the data for Machine Learning algorithms.
4. Select a model and train it.
5. Fine-tune your model (optional): 5 point bonus.

Getting the data: Get your data form (https://archive.ics.uci.edu/ml/datasets/Auto+MPG) there you can find also detail description of the dataset. Here, you can create a function to get the data or just download it and then upload it to your Jupyter notebook.

Discover and visualize the data to gain insights: Use a visual representation of your data to try to make sense of it. Histograms per attribute are good way to see how the data is distribute it. Look for correlations, correlate each feature to your response variable.

Prepare the data for Machine Learning algorithms: Fortunately, this data doesn't have missing values, however, evaluate the possibility of scaling or normalizing your data. Check the types of attributes you have, in your case :

1. mpg: continuous
2. cylinders: multi-valued discrete
3. displacement: continuous
4. horsepower: continuous
5. weight: continuous
6. acceleration: continuous
7. model year: multi-valued discrete
8. origin: multi-valued discrete
9. car name: string (unique for each instance)

Evaluate which will be the best way to encode attributes such as car name, cylinders, model ear, origin. Also evaluate the possibility of remove some attributes.

Create a training a testing set: Use your Sklearn libraries for this purpose, some option includes, train_test_split, and StratifiedShuffleSplit. Justify the use of using either one.

Train and evaluate several models (hypothesis functions): This include the selection of some models, the selection of cost function, and an optimization algorithm. Sklearn really simplify this stage for us.

Try the following models and use the following two approaches for training and testing

Approach 1: Here, use the training set to train the models, and make predictions using the testing set. For this part let's stick to distribution 80% of training data and 20% of testing.

- LinearRegression
- DecisionTreeRegressor
- RandomForestRegressor.

Use root mean square error as a cost function.

$$\text{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

Use gradient decent as a optimization algorithm

Approach 2: Here, we will use the same settings (models, cost function, and optimization algorithm). However, we will use 10-fold cross-validation, instead of the 80%, 20% of training-testing distribution approach.

Compare and analyses your results.

Remember:

Don't use your testing data for your training, this will cause overfitting or lack of generalization (check for that concept in chapter 1)

Evaluate your model using your testing set by following the next procedure

a. Make predictions using the attributes of your testing set.
b. Compare these predictions with the associated response variable (labels in classification) to these features
c. Compute the root mean square.

The code in python looks like that.

```
final_predictions = final_model.predict(X_test_prepared)

final_mse = mean_squared_error(y_test, final_predictions)
final_rmse = np.sqrt(final_mse)   # => evaluates to 47,766.0
```

Optional:

Read about Grid Search, method to find the best combination of hyper-parameters. Try this approach to find the best combination of hyper-parameters for RandomForestRegressor:.