CSCI567: Machine Learning	USC, Fall 2022
Homework 4	
Instructor: Vatsal Sharan	*Due: November 17 by 2:00 pm PST*

A reminder on collaboration policy and academic integrity: Our goal is to maintain an optimal learning environment. You can discuss the homework problems at a high level with other groups, but you should not look at any other group's solutions. Trying to find solutions online or from any other sources for any homework or project is prohibited, will result in zero grade and will be reported. To prevent any future plagiarism, uploading any material from the course (your solutions, quizzes etc.) on the internet is prohibited, and any violations will also be reported. Please be considerate, and help us help everyone get the best out of this course.

Please remember the Student Conduct Code (Section 11.00 of the USC Student Guidebook). General principles of academic honesty include the concept of respect for the intellectual property of others, the expectation that individual work will be submitted unless otherwise allowed by an instructor, and the obligations both to protect one's own academic work from misuse by others as well as to avoid using another's work as one's own. All students are expected to understand and abide by these principles. Students will be referred to the Office of Student Judicial Affairs and Community Standards for further review, should there be any suspicion of academic dishonesty.

Notes on notation:

- Unless stated otherwise, scalars are denoted by small letter in normal font, vectors are denoted by small letters in bold font and matrices are denoted by capital letters in bold font.
- $\|.\|$ means L2-norm unless specified otherwise, *i.e.*, $\|.\| = \|.\|_2$.

Instructions

We recommend that you use LaTeX to write up your homework solution. However, you can also scan handwritten notes. The homework will need to be submitted on Gradescope.

Theory-based Questions

Problem 1: Decision Trees (12pts)

Consider a binary dataset with 400 examples, where half of them belong to class A and the rest belong to class B. Next, consider two decision stumps (i.e. trees with depth 1) $\mathcal{T}_1$ and $\mathcal{T}_2$, each with two children. For $\mathcal{T}_1$, the left child has 150 examples in class A and 50 examples in class B. For $\mathcal{T}_2$, the left child has 0 examples in class A and 100 examples in class B. (You can infer the number of examples in the right child using the total number of examples.)

1.1 (6 pts) In class, we discussed entropy and Gini impurity as measures of uncertainty at a leaf. Another possible metric is the classification error at the leaf, assuming that the prediction at the leaf is the majority class among all examples that belong to that leaf. For each leaf of $\mathcal{T}_1$ and $\mathcal{T}_2$, compute the entropy (base e), Gini impurity and classification error. You can either exactly express the final numbers in terms of fractions and logarithms, or round them to two decimal places.

1.2 (6 pts) Compare the quality of $\mathcal{T}_1$ and $\mathcal{T}_2$ (that is, the two different splits of the root) based on conditional entropy (base e), weighted Gini impurity and total classification error. Intuitively, which of $\mathcal{T}_1$ or $\mathcal{T}_2$ appears to be a better split to you (there may not necessarily be one correct answer to this)? Based on your conditional entropy, Gini impurity and classification error calculations, which of the metrics appear to be more suitable choices to decide which variable to split on?

Problem 2: Gaussian Mixture Model and EM

This question will be released later.

Programming-based Questions

As in previous homeworks, you need to have your coding environment setup for this part. We use python3 (version $\geq$ 3.7) in our programming-based questions. There are multiple ways you can install python3, for example:

- You can use conda to configure a python3 environment for all programming assignments.
- Alternatively, you can also use virtualenv to configure a python3 environment for all programming assignments

After you have a python3 environment, you will need to install the following python packages:

- numpy
- matplotlib (for plotting figures)

Note: You are **not allowed** to use other packages such as *tensorflow, pytorch, keras, scikit-learn, scipy*, etc. for 3.1-3.2. If you have other package requests, please ask first before using them. You are **allowed** to use any packages for 3.3-3.5.

Download the files for the programming part from `https://vatsalsharan.github.io/fall22/hw4.zip`.

Problem 3: Exploring Decision Trees and Random Forests (12pts)

In this question, we will observe the effect of different hyperparameters in training decision trees and random forests, and also visualize what features the random forest model is using to make predictions. We will do this on a Colab notebook `HW4-Exploring-Random-Forests.ipynb`.

Instructions to run the notebook: Upload this file to your USC Google Drive. Then, add Google Colab to your Google App by New $\rightarrow$ More $\rightarrow$ Connect more apps $\rightarrow$ Type in Google Colab and install it, as in Fig. 1. After that, you can run the notebook with your browser.

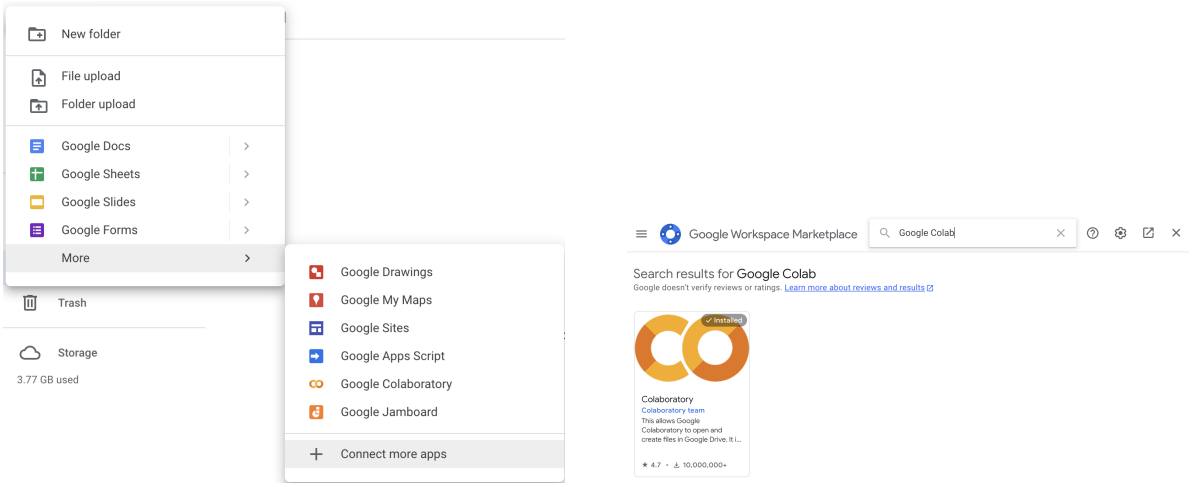


Figure 1: Screenshots showing how to install Colab.

We use a variant of the MNIST dataset for this problem. As mentioned in HW 3, the MNIST dataset contains images of handwritten digits (0, 1, 2, ..., 9) and is generally used for the (10) digit classification problem. Here, we work with a binary classfication task of predicting whether the digit is less than 5 or not. Fig. 2 shows some samples images from the dataset with original and binary labels, respectively.

You should go through the code we provide and understand what's happening, but for the purpose of answering the questions you will mainly have to run the code and understand the results. All the models (decision trees, random forests, etc.) are imported from the `sklearn` library. You should feel free to explore the role of other hyperparameters and other methods (such as bagging, boosting), and go through the documentation to better understand these things, but for this question, we will focus on decision trees and random forests.

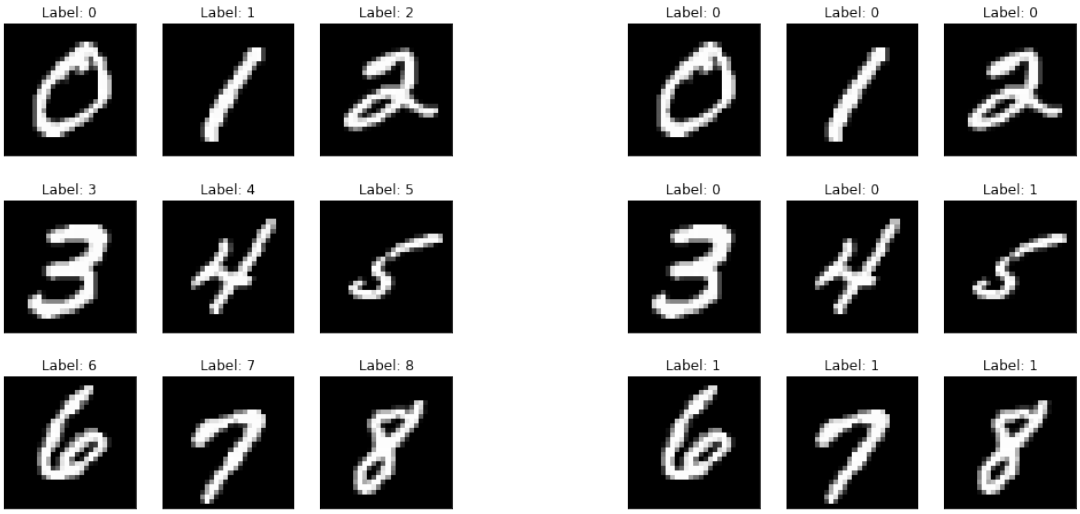


Figure 2: Some samples images from the MNIST dataset with original (left) and binary (right) labels, respectively.

We will look at the effect of 5 parameters:

- `max_depth`: The maximum depth of the decision tree.
- `n_estimators`: The number of decision trees in the forest.
- `min_samples_leaf`: The minimum number of samples required to be a leaf node.
- `max_samples`: The number of samples to draw from the training set to train each decision tree in the forest.
- `max_features`: The number of features to consider when looking for the best split for any node.

To observe the effect of a parameter, we look at train and test accuracy for different values of that parameter while keeping the rest of the parameters fixed.

3.1 (2 pts) The first set of plots shows the train and test accuracy for different values of `max_depth` using a decision tree and a random forest, respectively. How do the accuracies and the generalization gap vary with `max_depth` in these cases? For a particular value of `max_depth`, how does the generalization gap for the decision tree compare with that of the random forest? Explain your observations.

3.2 (2 pts) The next plot shows the train and test accuracy for different values of `n_estimators` for a random forest. Comment on your observations regarding the accuracies and the generalization gap. What value(s) (range or approximate values are enough) would you prefer to use for this parameter? Give reasons why. Hint: Are there any drawbacks to using very high values of `n_estimators`?

For the next three parts of this question, we will also look at the how the size of the training set influences the accuracy trends for a given parameter. In each case, for the first plot, the training set consists of 4000 samples whereas for the second plot, it contains 1000 samples.

3.3 (2 pts) The next set of plots shows the train and test accuracy for different values of `min_samples_leaf` for a random forest. Taking into account the behaviours for different training set sizes, explain your observations for very low and very high values of `min_samples_leaf`. What do you conclude from this trend? What could be the reasons for such a behaviour?

3.4 (2 pts) The next set of plots shows the train and test accuracy for different values of `max_samples` for a random forest. What do you observe for very low and very high values of `max_samples`? Would you prefer to use

low, intermediate or high values for this parameter in both the cases? Give reasons why. Hint: How does the size of the training set influence the choice of this parameter?

3.5 (2 pts) The next set of plots shows the train and test accuracy for different values of `max_features` for a random forest. Comment on your observations regarding the accuracies and the generalization gap for the two training set sizes. What is the best range of values for this parameter in both the cases? Is it similar/different? Explain your observations.

3.6 (2 pts) In class, we discussed how ensembles are usually not as interpretable as a single decision tree. While this is true, there are still ways to explore which features are used the most by our ensemble. We will explore one such technique in this part.

We visualize the *feature importances* of a random forest model trained for the binary classification task on the MNIST dataset. Intuitively, features with higher importance are the pixels which are used more often in the decision trees in the forest and which lead to better splits, i.e. which contribute more in improving the performance of the model. For more details, you can see Section 18.6.1 of the PML book. The last plot shows the importances of different pixels/portions of the image for a trained random forest model to make its predictions. What portions of the image does the model seem to be focusing on? In other words, can you think of reasons why the pixels with higher importance are indeed important for the prediction task (classifying whether the digit is smaller than 5 or not)? As is usually true for such open-ended questions, there can be multiple correct answers here and we're looking more for your reasoning than a specific answer.

Problem 4: PCA for Learning Word Embeddings

This question will be released later.

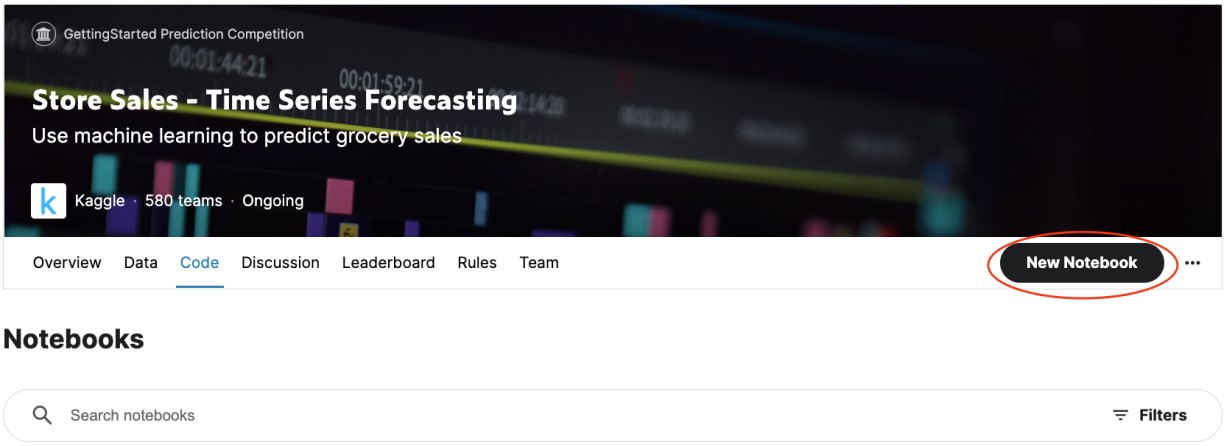
Problem 5: Project Launchpad (16pts + 10pts bonus)

In this part, you will implement a simple linear regression model for the final project. In the meantime, you will also explore the use of the Pandas package, a commonly used library for Kaggle competition (Note: it does not mean that you have to use Pandas for your project. Here is a tutorial for the Pandas package).

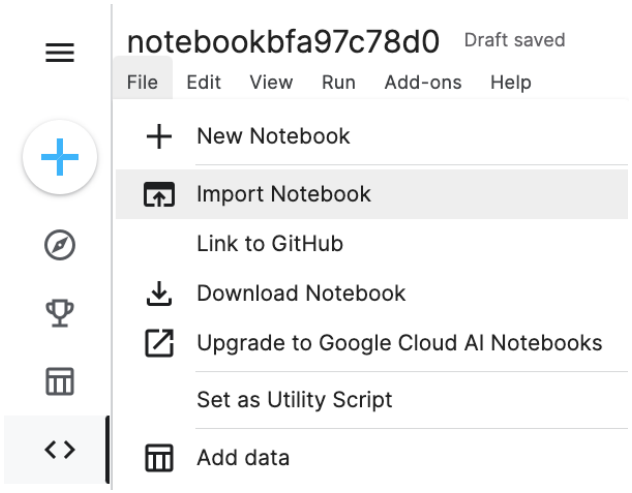
Early bird bonus We encourage you to start early on the project. The first 5 teams among all teams in CSCI-567 class at the **HW4 submission deadline** can get **10 bonus points**. To get this bonus, at least one of the team members will have to give a 5-minute presentation on their approach in the discussion session on November 17. We will consider the final ranking on the public leaderboard as of **2pm PST on Nov 16** for this part of bonus points.

Setup We will complete this part of assignment on a jupyter notebook `project-starter.ipynb`. To load the notebook into Kaggle, follow these steps:

(a) In the "Code" tag of the competition, click button "New Notebook".

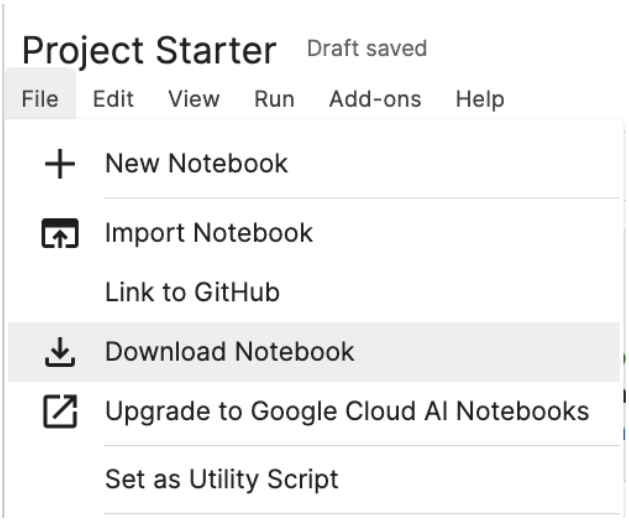


(b) Then, select "File/Import Notebook", and import `regression.ipynb` into Kaggle.



Deliverables Notebook with completed code, output, and the discussion for 5.4.

Submission Instruction Please submit your notebook *along with the output* for grading. If you are using Kaggle notebook, you can export the notebook by "File/Download Notebook", see figure below:



General Tip 1: before getting started, please read the dataset description carefully. Make sure you understand the datatype of each entries and the meanings behind them.
General Tip 2: all of the tasks in this part can be done within three lines of code. You should spend most of your time reading documentations to understand certain methods provided by the Pandas package, instead of implementing the functionalities yourself.

Part I - Feature Engineering (12 pts)

One of the most important steps in developing a machine learning model is feature engineering. In this part, we will walk you through the process of constructing a simple set of features for the final project.

5.1 Moving average of oil prices (6 pts) In this task, you are asked to compute the moving average of the oil prices with window size 7.
5.1.1 (2 pts) In Section 1 of the notebook, after loading the oil price into `data_oil`, compute `data_oil["ma_oil"]` as the moving average of past seven recorded oil prices (note that the oil prices of some dates are missing, and we will skip them in the computation).
5.1.2 (4 pts) After computing `data_oil["ma_oil"]`, you should now deal with the missing dates in `data_oil["ma_oil"]`. A DataFrame `calendar` with contiguous dates from 2013-01-01 to 2017-08-31 is created for you. You are asked to complete the following:

(a) Merge DataFrames `calendar` and `data_oil`. The merged DataFrame should have contiguous dates from 2013-01-01 to 2017-08-3 as indexes, and the same columns as `data_oil`, where missing values are denoted by `NaN`.

(b) Fill in the missing values of `data_oil["ma_oil"]`. There are many ways to deal with missing values, such as fill it with some default value, fill it with the last valid observation, or fill it with the next valid observation. Here, you are asked to fill in each missing value by the last valid observation. Perform this operation *in place* on `data_oil["ma_oil"]`.

5.2 Workday feature (3 pts) Next, we will create an extra feature indicating whether each date is a workday. Here, we only make use of the fact that Monday to Friday are usually workdays, and we ignore all the holidays information in `holidays_events.csv` (which is left for you to explore). In Section 2 of the notebook, your task is to create a column `calendar["wd"]` such that a date between Monday and Friday has value `True`, and a date that is Saturday or Sunday has value `False`.

5.3 Create trend feature (3 pts) Now we are ready to load all training and testing data into the notebook (Section 3). The last step before training our model is to create the trend feature: note that a linear regression model cannot read the dates, and we need to map the dates into certain features. Here, you are asked to implement the simplest trend feature: given a range of dates, map them into a range of integers starting from 1. You need to create a DataFrame `X` as the trend feature, which is then extended with the moving average of oil prices and workday indicators to form the final training features.

Part II - Train Your Model (4 pts)

We are now ready to train the model! Here, we will train a simple linear regression model, which only takes a few seconds.

5.4 Display training error by the type of product (2 pts) Let's show the training error to get a sense of how well the model is fitting the training data. The model is making sales predictions for each store and type of product. You are asked to show the training error within each type of product. Which family of product does the model has the most difficulty in predicting the sales?

5.5 Make predictions on the test set (2 pts) Now you can use your trained model to make predictions on the test set. To do this, you need to first create the features for the test set (denoted by `X_test`) similar to the training set. Fill in your steps for computing `X_test` in the notebook.

5.6 Make a submission You are now ready to make a submission. Simply open the "Competitions" tag on the right, and click the submit button to make the submission. You should get a score around 0.45. This question will be released later.