Project 4: Regression Analysis and Define Your Own Task!

Due on Mar 14, 2025, 11:59 pm

1 Introduction

Regression analysis is a statistical procedure for estimating the relationship between a target variable and a set of features that jointly inform about the target. In this project, we explore specific-to-regression feature engineering methods and model selection that jointly improve the performance of regression. You will conduct different experiments and identify the relative significance of the different options.

2 Datasets

You should take steps in section 3 on **either one of** the following datasets.

2.1 Dataset 1: Diamond Characteristics

Valentine's day might be over, but we are still interested in building a bot to predict the price and characteristics of diamonds. A synthetic diamonds dataset can be downloaded from this **link**. This dataset contains information about 150000 round-cut diamonds. There are 14 variables (*features*) and for each sample, these features specify the various properties of the sample. Below we describe some of these features:

- **carat**: weight of the diamond
- **cut**: quality of the cut
- **clarity**: measured diamond clarity
- **length**: measured length in mm
- **width**: measured width in mm
- **depth**: measured depth in mm
- **depth_percent**: diamond's total height divided by it's total width
- **table_percent**: width of top of diamond relative to widest point
- **gridle_min**: refers to the thinnest part of the girdle
- **gridle_max**: refers to the thickest part of the girdle

In addition to these features, there is the target variable: i.e what we would like to predict:

- **price**: price in US dollars

2.2 Dataset 2: Wine Quality Dataset

Perhaps your interests lean more towards the nuances of wine tasting rather than the than the fascination with diamonds. You can access the dataset through this **link**. The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. This dataset comprises $4,898$ instances for white wine and 1599 instances for red wine, with 13 features for each instance.

3 Required Steps

In this section, we describe the setup you need to follow. Follow these steps to process ***either of the*** datasets in Section 2.

3.1 Before Training

Before training an algorithm, it's always essential to inspect the data. This provides intuition about the quality and quantity of the data and suggests ideas to extract features for downstream ML applications. In this following section we will address these steps.

3.1.1 Handling Categorical Features

A categorical feature is a feature that can take on one of a limited number of possible values. If one dataset contains categorical features, a preprocessing step needs to be carried to convert categorical variables into numbers and thus prepared for training.

One method for numerical encoding of categorical features is to assign a scalar. For instance, if we have a "Quality" feature with values {`Poor, Fair, Typical, Good, Excellent`} we might replace them with numbers 1 through 5. If there is no numerical meaning behind categorical features (e.g. {`Cat, Dog`}) one has to perform "one-hot encoding" instead.

3.1.2 Data Inspection

The first step for data analysis is to take a close look at the dataset[1].

- Plot a heatmap of the Pearson correlation matrix of the dataset columns. Report which features have the highest absolute correlation with the target variable. **In the context of either dataset, describe what the correlation patterns suggest.** **Question 1.1**
- Plot the histogram of numerical features. What preprocessing can be done if the distribution of a feature has high skewness? **Question 1.2**
- Construct and inspect the box plot of categorical features vs target variable. What do you find? **Question 1.3**
- For the Diamonds dataset, plot the counts by color, cut and clarity. **or**
- For the wine quality dataset, plot histogram for quality scores. **Question 1.4**

[1] For exploratory data analysis, one can try pandas-profiling

3.1.3 Standardization

Standardization of datasets is a common requirement for many machine learning estimators; they might behave badly if the individual features do not more-or-less look like standard normally distributed data: Gaussian with zero mean and unit variance. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

Standardize feature columns and prepare them for training. **Question 2.1**

3.1.4 Feature Selection

- `sklearn.feature_selection.mutual_info_regression` function returns estimated mutual information between each feature and the label. Mutual information (MI) between two random variables is a non-negative value which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency.
- `sklearn.feature_selection.f_regression` function provides F scores, which is a way of comparing the significance of the improvement of a model, with respect to the addition of new variables.

You **__may__** use these functions to select features that yield better regression results (especially in the classical models). Describe how this step qualitatively affects the performance of your models in terms of test RMSE. Is it true for all model types? Also list two features for either dataset that has the lowest MI w.r.t to the target. **Question 2.2**

From this point on, you are free to use any combination of features, as long as the performance on the regression model is on par (or slightly worse) than the Neural Network model.

3.2 Training

Once the data is prepared, we would like to train multiple algorithms and compare their performance using average RMSE from 10-fold cross-validation (please refer to part 3.3).

3.3 Evaluation

Perform 10-fold cross-validation and measure average RMSE errors for training and validation sets.
For random forest model, measure "Out-of-Bag Error" (OOB) as well.

3.3.1 Linear Regression

What is the objective function? Train three models: (a) ordinary least squares (linear regression without regularization), (b) Lasso and (c) Ridge regression, and answer the following questions.

- Explain how each regularization scheme affects the learned parameter set. **Question 4.1**

- Report your choice of the best regularization scheme along with the optimal penalty parameter and explain how you computed it. **Question 4.2**
- Does feature standardization play a role in improving the model performance (in the cases with ridge regularization)? Justify your answer. **Question 4.3**
- Some linear regression packages return p-values for different features[2]. What is the meaning of these p-values and how can you infer the most significant features? A qualitative reasoning is sufficient. **Question 4.4**

3.3.2 Polynomial Regression

Perform polynomial regression by crafting products of features you selected in part 3.1.4 up to a certain degree (max degree 6) and applying ridge regression on the compound features. You can use `scikit-learn` library to build such features. Avoid overfitting by proper regularization. Answer the following:

- What are the most salient features? Why? **Question 5.1**
- What degree of polynomial is best? How did you find the optimal degree? What does a very high-order polynomial imply about the fit on the training data? What about its performance on testing data? **Question 5.2**

3.3.3 Neural Network

You will train a multi-layer perceptron (fully connected neural network). You can simply use the `sklearn` implementation:

- Adjust your network size (number of hidden neurons and depth), and weight decay as regularization. Find a good hyper-parameter set systematically (no more than 20 experiments in total). **Question 6.1**
- How does the performance generally compare with linear regression? Why? **Question 6.2**
- What activation function did you use for the output and why? You may use none. **Question 6.3**
- What is the risk of increasing the depth of the network too far? **Question 6.4**

3.3.4 Random Forest

We will train a random forest regression model on datasets, and answer the following:

- Random forests have the following hyper-parameters:
  - Maximum number of features;
  - Number of trees;
  - Depth of each tree;

  Explain how these hyper-parameters affect the overall performance. Describe if and how each hyper-parameter results in a regularization effect during training. **Question 7.1**

[2] E.g: `scipy.stats.linregress` and `statsmodels.regression.linear_model.OLS`

- How do random forests create a highly non-linear decision boundary despite the fact that all we do at each layer is apply a threshold on a feature? **Question 7.2**
- Randomly pick a tree in your random forest model (with maximum depth of 4) and plot its structure. Which feature is selected for branching at the root node? What can you infer about the importance of this feature as opposed to others? Do the important features correspond to what you got in part 3.3.1? **Question 7.3**
- Measure "Out-of-Bag Error" (OOB). Explain what OOB error and R2 score means. **Question 7.4**

3.3.5 LightGBM, CatBoost and Bayesian Optimization

Boosted tree methods have shown advantages when dealing with tabular data, and recent advances make these algorithms scalable to large scale data and enable natural treatment of (high-cardinality) categorical features. Two of the most successful examples are LightGBM and CatBoost.

Both algorithms have many hyperparameters that influence their performance. This results in large search space of hyperparameters, making the tuning of the hyperparameters hard with naive random search and grid search. Therefore, one may want to utilize "smarter" hyperparameter search schemes. We specifically explore one of them: Bayesian optimization.

In this part, **pick either one** of the datasets and apply LightGBM OR CatBoost. If you do both, we will only look at the first one.

- Read the documentation of LightGBM OR CatBoost and determine the important hyperparameters along with a search space for the tuning of these parameters (keep the search space small). **Question 8.1**
- Apply Bayesian optimization using `skopt.BayesSearchCV` from `scikit-optmize` to find the ideal hyperparameter combination in your search space. Keep your search space small enough to finish running on a single Google Colab instance within 60 minutes. Report the best hyperparameter set found and the corresponding RMSE. **Question 8.2**
- Qualitatively interpret the effect of the hyperparameters using the Bayesian optimization results: Which of them helps with performance? Which helps with regularization (shrinks the generalization gap)? Which affects the fitting efficiency? **Question 8.3**

Show Us Your Skills: Twitter Data

Introduction

As a culmination of the four projects in this class, we introduce this final dataset that you will explore and your task is to walk us through an end-to-end ML pipeline to accomplish any particular goal: regression, classification, clustering or anything else. **This is a design question and it is going to be about** 30% **of your grade in this project.** This is a chance to push yourself and be creative.

Below is a description and some small questions about the provided dataset to get you started and familiarized with the dataset:

3.4 About the Data

Download the training tweet data[3]. The data consists of 6 text files, each one containing tweet data from one hashtag as indicated in the filenames.

Report the following statistics for each hashtag, i.e. each file has: **Question 9.1**

- Average number of tweets per hour
- Average number of followers of users posting the tweets per tweet (to make it simple, we average over the number of tweets; if a users posted twice, we count the user and the user's followers twice as well)
- Average number of retweets per tweet

Plot "number of tweets in hour" over time for `#SuperBowl` and `#NFL` (a bar plot with 1-hour bins). The tweets are stored in separate files for different hashtags and files are named as `tweet_[#hashtag].txt`. **Question 9.2**

Note: The tweet file contains one tweet in each line and tweets are sorted with respect to their posting time. Each tweet is a JSON string that you can load in Python as a dictionary. For example, if you parse it to object `json_object = json.loads(json_string)`, you can look up the time a tweet is posted by:

`json_object['citation_date']`

You may also assess the number of retweets of a tweet through the following command:

`json_object['metrics']['citations']['total']`

Besides, the number of followers of the person tweeting can be retrieved via:

`json_object['author']['followers']`

The time information in the data file is in the form of UNIX time, which "encodes a point in time as a scalar real number which represents the number of seconds that have passed since the beginning of 00:00:00 UTC Thursday, 1 January 1970" (see Wikipedia for details). In Python, you can convert it to human-readable date by

```
import datetime
datetime_object = datetime.datetime.fromtimestamp(unix_time)
```

The conversion above gives out a `datetime` object storing the date and time in your local time zone corresponding to that UNIX time.

[3] https://ucla.box.com/s/24oxnhsoj6kpxhl6gyvuck25i3s4426d

In later parts of the project, you may need to use the PST time zone to interpret the UNIX timestamps. To specify the time zone you would like to use, refer to the example below:

```
import pytz
pst_tz = pytz.timezone('America/Los_Angeles')
datetime_object_in_pst_timezone =
 ↪  datetime.datetime.fromtimestamp(unix_time, pst_tz)
```

For more details about `datetime` operation and time zones, see
https://medium.com/@eleroy/10-things-you-need-to-know-about-date-and-time-in-pytho
Follow the steps outlined below: **Question 10**

- Describe your task.
- Explore the data and any metadata (you can even incorporate additional datasets if you choose).
- Describe the feature engineering process. Implement it with reason: Why are you extracting features this way - why not in any other way?
- Generate baselines for your final ML model.
- A thorough evaluation is necessary.
- Be creative in your task design - use things you have learned in other classes too if you are excited about them!

We value creativity in this part of the project, and your score is partially based on how unique your task is. Here are a few pitfalls you should avoid (there are more than this list suggests):

- DO NOTperform simple sentiment analysis on Tweets: running a pre-trained sentiment analysis model on each tweet and correlating that sentiment to the score in the game in time would give you an obvious result.
- DO NOT include trivial baselines: In sentiment analysis, for example, if you are going to try and train a Neural Network or use a pre-trained model, your baselines need to be competitive. Try to include alternate network architectures in addition to simple baselines such as random or naive Bayesian baselines.

Here we list a few project directions that you can consider and modify. These are not complete specifications. You are free and are encouraged to create your projects /project parts (that may get some points for creativity). The projects you come up with should match or exceed the complexity of the following 3 suggested options:

- **Time-Series Correlation between Scores and Tweets:** Since this tweet dataset contains tweets that were posted before, during, and after the Superbowl, you can find time-series data that have the real-time score of the football game as the tweets are being generated. This score can be used as a dynamic label for your raw tweet dataset: there is an alignment between the tweets and the score. You can then train a model to predict, given a tweet, the team that is winning. Given the score change, can you generate a tweet using an ensemble of sentences from the original data (or using a generative model that is more sophisticated)?

Figure 1: A sample of the significant events in the game that you can easily find on the internet. Here is one link that has the time-indexed events.

- **Character-centric time-series tracking and prediction:** In the #gopatriots dataset, there are several thousand tweets mentioning "Tom Brady" and his immediate success/failure during the game. He threw 4 touchdowns and 2 interceptions, so fan emotions about Brady throughout the game are fickle. Can we track the average perceived emotion across tweets about each player in the game across time in each fan base? Note that this option would require you to explore ways to find the sentiment associated with each player in time, not to an entire tweet. Can we correlate these emotions with the score and significant events (such as interceptions or fumbles)? Using these features, can you predict the MVP of the game? Who was the most successful receiver? The MVP was Brady.
- **Library of Prediction Tasks given a tweet:** Predict the hashtags or how likely it is that a tweet belongs to a specific team fan. Predict the number of retweets/likes/quotes. Predict the relative time at which a tweet was posted.

Some caveats:

- The dataset is large. Feel free to subsample as needed.
- The dataset is highly imbalanced with respect to fan distribution. Please take that into account during your analysis.
- Few useful and trending SW libraries you might consider but are not required to use: Pinecone, Guardrails, Llama, BERT, PySpark/Spark, PyG.
- Though we provide sufficient data to apply Deep Learning to the tasks, you are not required to do so. On the contrary if you do apply DL and fail to show all the proof points (convergence, qualitative evaluation), you will lose points.

Submission

Your submission should be made to both of the two places: BruinLearn and Gradescope within BruinLearn.

BruinLearn Please submit a zip file containing your **report**, and your **codes** with a **readme file** on how to run your code to BruinLearn. The zip file should be named as

"Project1_UID1_UID2_..._UIDn.zip"

where UIDx's are student ID numbers of the team members. **Only one submission per team is required**. If you have any questions, please ask on Piazza or through email. Piazza is preferred.

Gradescope Please submit your report to Gradescope as well. Please specify your group members in Gradescope. It is very important that you assign each part of your report to the question number provided in the Gradescope template.