CSC 635 Data Mining

## Final Project Report

### Submitted to:

### Dr. Jamil Saquer

### Author(s):

### Sharmin Sultana

### Sam Nack

**Abstract**

Assessing a professor is a difficult and subjective task. Here, we employ a vast corpus of student evaluations gathered from the RateMyProfessors website [1], which spans a variety of institutions, fields, and cultures. Our task is to predict the ratings of a professor based on the tags students marked for that faculty. We built prediction models by using 6 popular regression algorithms (linear regression, K-Nearest Neighbours regression, Support Vector Regression, Decision Tree Regression, Random Forest Regression, and Multi-layer Perceptron Regression) and evaluated their performances by analysing Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

**Introduction**

The RateMyProfessor website allows students to leave ratings for college professors. The purpose of this site is to give students access to information on their instructors before they register for classes. This gives students the information they need to be able to pick their courses wisely. Each review written by students can have multiple components. The main parts of a review include: an open text feedback, a difficulty rating, a star rating, and the option to select various “tags” to describe the professor. The site states that it has over 19 million ratings across 1.7 million professors and 7,500 schools. For this project, we used a dataset collected from RateMyProfessor. The dataset used a program to collect data from the site. The data was downloaded as a csv file with 20,000 records. Each record corresponded to a single student rating and included 51 features. Most of these features were discarded for our project, included in Table 1 is a list of the features used.

|  |  |
| --- | --- |
| **Column Name** | **Brief Description** |
| professor\_name | The name of the professor being rated |
| star\_rating | The average star rating of the professor, based on all student reviews. A decimal from 1.0 (low) to 5.0 (high) |
| gives\_good\_feedback  caring  respected  participation\_matters  clear\_grading\_criteria  skip\_class  amazing\_lectures  inspirational  tough\_grader  hilarious  get\_ready\_to\_read  lots\_of\_homework  accessible\_outside\_class  lecture\_heavy  extra\_credit  graded\_by\_few\_things  group\_projects  test\_heavy  so\_many\_papers  beware\_of\_pop\_quizzes  IsCourseOnline | Binary, is 1 if this is one of the common tags of the professor being rated, is 0 otherwise. |

*Table 1. Features in the Dataset*

For this project, we decided to attempt to regress the professor’s average star rating based on their popular tags. This would not take into account individual reviews, and instead focus on the last 21 features of the dataset, which are the binary attributes that describe the popular tags of the professor. This brought our total number of records from 20,000, where each record indicated a single review, to about 626, where each record would represent one professor. We wanted to do this because the star rating is one of the key metrics in determining how to choose professors. In fact, on the site itself, the preview for each professor is their star rating and their name listed. We decided to focus on regression using the tags for several reasons. Firstly, it is not obvious whether many of the features of the dataset are relevant. For example, how long a specific instructor has been teaching does not seem as relevant as whether a professor could be described as “caring”. Secondly, the abundance of the tags feature in the dataset made it easier to analyse. There are 21 tags, so this gives a decent number of features for regression. Also, there are only around 9000 records in the dataset that have null or missing values for this tags features, whereas there are over 15,000 missing values for some of the other features such as “take\_agains”, “grades”,”attendance” and others.

**Background:**

The study reported in this project is based on a large corpus of student evaluations collected from the RateMyProfessors website [1], covering different institutions, disciplines, and cultures. Our goal was to predict the ratings of a professor based on 21 tags that students choose for that professor.

Our first task was to decide what type of analysis (regression or classification) we are going to do with the dataset. Since our target was to estimate the ratings of a professor based on the student’s evaluations (tags) which are continuous numeric values, this is a regression analysis problem.

There are several types of regression algorithms. In this project, we built models based on 6 regression algorithms (Linear Regression, K-Nearest Neighbours (KNN) Regression, Support Vector Regression (SVR), Decision Tree Regression, Random Forest Regression, and Multi-layer Perceptron Regression) and compared performance of the models.

**1. Linear Regression:** This is the simplest regression model, in which the input variables (x) and the single output variable (y) are assumed to have a linear relationship (y). Since there are more than one value of x (21 tags) in our dataset, it is basically multiple linear regression type problem.

Chart, scatter chart

Description automatically generated

*Figure 1: Pictorial representation of multiple linear regression [2]*

The red line in the above graph is referred to as the best fit straight line. Based on the given data points, we try to plot a line that models the points the best. This best fit line is drawn by using following formula.

**2. K-Nearest Neighbours (KNN) Regression:** The KNN algorithm uses ‘feature similarity based on distances between two points’ to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set.

* 1. The working procedure of KNN is described below:

1. Load the data
2. Initialize K which defines the chosen number of neighbours
3. For each example in the data
   1. Calculate the distance between the query example and the current example from the data. Distances are calculated using Minkowski, Euclidean or Manhattan distance formula. By default, distances are calculated using Minkowski.
   2. Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels.

Chart, scatter chart

Description automatically generated

*Figure 2: Pictorial representation of KNN regression algorithm [3]*

**3. Support Vector Regressor (SVR):** The equation of the line in Support Vector Regression is , which is comparable to Linear Regression. This straight line is known as hyperplane in SVR. The nearest data points on either side of the hyperplane are termed Support Vectors, and they are used to plot the boundary line.

Unlike other regression models, the SVR aims to fit the best line within a threshold value (distance between hyperplane and boundary line), rather than minimizing the error between the real and predicted value. As a result, we can state that the SVR model attempts to satisfy the criteria . To anticipate the value, it used the locations along this boundary.

Chart

Description automatically generated

*Figure 3: Pictorial representation of SVR [4]*

**4. Decision Tree Regressor:** In the shape of a tree structure, a decision tree constructs regression or classification models. It incrementally cuts down a dataset into smaller and smaller sections while also developing an associated decision tree. A tree with decision nodes and leaf nodes is the result. A decision node can have two or more branches, each of which represents a value for the attribute being checked. A decision on the numerical aim is represented as a leaf node. The root node is the topmost decision node in a tree that corresponds to the best predictor. Both category and numerical data can be handled by decision trees.

The primary approach for creating decision trees is a top-down, greedy search across the universe of possible branches that does not require backtracking. By substituting Standard Deviation Reduction for Information Gain, we can calculate the homogeneity of a numerical sample to build a decision tree for regression.

The decision tree regression algorithm works in the following manner:

1. The standard deviation of the target is calculated.
2. The dataset is then split on the different attributes. The standard deviation for each branch is calculated. The resulting standard deviation is subtracted from the standard deviation before the split. The result is the standard deviation reduction.
3. The attribute with the largest standard deviation reduction is chosen for the decision node.
4. The dataset is divided based on the values of the selected attribute. This process is run recursively on the non-leaf branches, until all data is processed.
5. When the number of instances is more than one at a leaf node we calculate the average as the final value for the target.

A picture containing diagram

Description automatically generated

*Figure 4: Demo representation of decision tree regression algorithm [5]*

**5. Random Forest Regressor:** Random Forest is a Supervised Learning technique for classification and regression that employs the ensemble learning method. The trees in random forests are run in parallel. There is no interaction between these trees while building the trees. It fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max\_samples parameter.

**Diagram

Description automatically generated**

*Figure 5: Random Forest structure [6]*

**6**. **Multilayer Perceptron Regression:** The Multi-layer Perceptron (MLP) is a supervised learning technique that trains on a dataset to learn a function. It can learn a non-linear function approximator for classification or regression given a set of features and a target. It differs from logistic regression in those one or more non-linear layers, known as hidden layers, can exist between the input and output layers**.**

Here, a short description is given how perceptron algorithm works.

1. At the very first step, generate k random weights equal to (n+1). Here, n is the number of feature columns in the dataset. For example, if one is given a dataset of three (3) features then s/he needs to generate four (4) random weights. The extra weight i.e., weight [0] is used for bias. Bias weight is an adjustable, numerical term used for increasing the classification model accuracy.

For each training sample

1. We need to calculate the weighted sum of the perceptron’s weights and input value. The default input value for bias weight is 1.

[Equation 1]

1. In this step, error is calculated by subtracting the values of from the expected class label which is y.

[Equation 2]

1. If error is zero (0), do nothing else we need to update next weights based on the following equation:

[Equation 3]

Here, defines the learning rate which should be in between [0,1]. Smaller learning rate tends to make the perceptron more stable and noise resistant.

1. Follow steps (II – IV), until stopping condition is fulfilled. Stopping condition means total number of iterations which is also known as epoch. It indicates how long this training process will continue.
2. After certain number of iterations, we need to set a threshold value to predict the class label for the test dataset. Usually step/threshold activation function is used to classify the objects.

[Equation 4]

Here,

**Evaluating the Algorithms:** This step compares how well different algorithms perform on a particular dataset. For regression algorithms, three evaluation metrics are commonly used:

* **Mean Absolute Error (MAE)** is the mean of the absolute value of the errors. It is calculated as:
* **Mean Squared Error (MSE)** is the mean of the absolute squared value of the errors. It is calculated as:
* **Root Mean Squared Error (RMSE)** is the square root of the mean of the squared errors.

**Implementation**

**Preprocessing:** This step loads the dataset and prepares it for regression. To prepare the dataset so that we could work with it, we used pandas to read the data in from the csv file. We did some preliminary analysis of the dataset, including looking at the statistics of the dataframe. Then we dropped all of the columns of the dataframe except the professor’s name, star rating, and the binary tags described in Table 1 above. We used the professor’s name to group all the ratings of a single professor and average the star rating and tag values. Next, we normalized the professor’s star rating to a range of 0 to 1 instead of 1 to 5. We then split the dataset into a training and testing set using sklearn’s train\_test\_split function. We decided to use 10% of the data for testing and the remaining 90% for training.

**Algorithm Implementation:** To build our models based on 6 regression algorithms, we used built-in machine learning library scikit-learn for the python programming language. For each algorithm, scikit-learn has specific module and library. After importing those modules, we fitted our training dataset into the model. Once training is done, we used *predict()* method to estimate output for the test dataset (independent features). With this predicted value, we evaluate our algorithm’s efficiency by calculating MAE, MSE, and RMSE. In scikit-learn, there is a module named metrics that has all the built-in functions to calculate the errors (MAE, MSE, RMSE).

In the following table, modules used for each algorithm are listed:

|  |  |
| --- | --- |
| Algorithm | Modules in scikit-learn |
| Linear Regression | from sklearn.linear\_model import LinearRegression |
| K-Nearest Neighbours Regression | from sklearn.neighbors import KNeighborsRegressor |
| Support Vector Regression | from sklearn.svm import SVR |
| Decision Tree Regression | from sklearn.tree import DecisionTreeRegressor |
| Random Forest Regression | from sklearn.ensemble import RandomForestRegressor |
| Multi-layer Perceptron Regression | from sklearn.neural\_network import MLPRegressor |

*Table 2: Libraries and modules used for each regression analysis*

**Results**

After running all of our regression methods, we calculated the MAE, MSE, and RMSE of each method and printed them out. We summarized these results in bar graphs.

**Icon

Description automatically generated**

*Figure 6: Comparison of MAE of all regression techniques*

We can see from this graph that the decision tree had the highest mean absolute error and that the linear regression model had the lowest.

**Icon

Description automatically generated**

*Figure 7: Comparison of MSE of all regression techniques*

We can see from this graph that the decision tree had the highest mean squared error and that the linear regression and support vector machine models had the lowest.

Icon

Description automatically generated

*Figure 8: Comparison of RMSE of all regression techniques*

We can see from this graph that the decision tree had the highest root mean squared error and that the linear regression and support vector machine models had the lowest.

**Conclusion**

Our results obviously indicated that the decision tree was the least viable model for this regression. The SVM and Linear Regression techniques performed admirably on our data. Just based on the value of a few tags, they were able to get decent predictions that were within a few tenths of a point of the true value.

In the future, this work could be expanded upon. For example, different distance metrics for KNN could be used that better model the relationship between tags. We could also try to modify the SVM kernel to better suit this specific domain. A deep learning approach could also be employed. We could also attempt to do textual analysis of the individual comments. Or, we could also try and determine what departments or colleges get higher or lower ratings than others.

Even given just a few tags, the models we created here were able to predict well on the test data. This shows that there is most likely some sort of correlation between the tags and the rating of the professor. Students can use this information to glean more knowledge from each rate my professor page. Instructors and universities could also use this to try and determine what behaviours students value or dislike in their instructors. This information could also be applied to other domains that include tags in their rating system.

**Member Contributions**

We are both satisfied with the work completed by each team member. Here we have listed out the specific contributions to the codebase of all team members.

**Sam:**

Sam implemented the pre-processing component of the code. He also implemented the Multi-layer perceptron, K-nearest neighbours, and decision tree models. Finally, he utilized the various metrics calculated on each model to generate a graphical representation of the performances of the different models across all metrics.

**Sharmin**:

Sharmin implemented the Linear Regression, Support Vector Regression, and Random Forest Regression. Finally, she evaluated each model’s performance by calculating 3 types of error metrics that are MAE, MSE, and RMSE.

**References**

|  |  |
| --- | --- |
| [1] | He, J. (2020, March 4) “*Big Data Set from RateMyProfessor.com for Professors” Teaching Evaluation*. Mendeley Data, V2.<https://data.mendeley.com/datasets/fvtfjyvw7d/2> |
| [2] | Dash, D. (2021, February 26). “*Multiple Linear Regression from scratch using only numpy*. “Medium.<https://medium.com/analytics-vidhya/multiple-linear-regression-from-scratch-using-only-numpy-98fc010a1926> |
| [3] | *“KNN Classification using Sklearn Python*. “, DataCamp Community.  <https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn> |
| [4] | K, G. M. (2020, July 18). “*Machine Learning Basics: Support Vector Regression - Towards Data Science*.” Medium. <https://towardsdatascience.com/machine-learning-basics-support-vector-regression-660306ac5226> |
| [5] | Dubey, B. (2020, January 11). “*Decision Tree Regression - Bhartendu Dubey*.”  Medium. <https://medium.com/@bhartendudubey/decision-tree-regressione202008c2df> |
| [6] | Chakure, A. (2020, November 6). “*Random Forest Regression - The Startup”*. Medium. <https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f> |