| | |
|---|---|
| Internship Project Title | TCS iON RIO-210: Salary Prediction Dashboard for HRs |
| Name of the Company | TCS iON |
| Name of the Industry Mentor | Avinash Singh |
| Name of the Institute | Amity University Online |

| Start Date | End Date | Total Effort (hrs.) | Project Environment | Tools used |
|---|---|---|---|---|
| 02/12/2024 | | 85 hrs | Jupyter notebook | Python 3 |

| | | | |
|---|---|---|---|
| Milestone # | 1 | Milestone : | Create, clean, and sanitize the dataset. Train the dataset and predict the salary of a particular HR based on it. |

**TABLE OF CONTENT**



## Acknowledgments

I, Archita Debnath, a student pursuing a Master of Computer Applications with a specialization in Machine Learning, hereby declare that I have prepared My Internship Project titled, **"TCS iON RIO-210: Salary Prediction Dashboard for HRs,"** under the guidance of Mr. Avinash Singh.

First and foremost, I sincerely express my gratitude to my industry mentor, Mr. Avinash Singh, for his valuable support, guidance, and encouragement throughout this project. His expertise and insights were crucial in helping me achieve the objectives of this work. I am deeply thankful for his availability and for patiently addressing my queries at every phase of the project.

## Objective

The objective of this project is to train machine learning models to accurately predict salaries and to develop a user-friendly, efficient salary prediction dashboard specifically designed for HR professionals.

## Introduction

During the first 7 days of my project, I collected and cleaned the dataset, ensuring it was properly cleaned, sanitized and ready for training. The dataset is now prepared for use in salary prediction. Over the next 15 days, I conducted exploratory data analysis by visualizing the relationship between various attributes and the salary of HR professionals. Subsequently, I trained the dataset using machine learning models, including Logistic Regression, Random Forest, and Support Vector Machine (SVM). Additionally, I performed hyperparameter tuning to optimize model performance. After that, I have generated the classification report. Finally, I have predicted the result using a user-defined data tuple.

## Internship Activities

- Watched all welcome kit videos
- Did preparation for RIO – pre-assessment test
- Attended the RIO – pre-assessment test
- Gone through the day-wise plan
- Read the project reference material
- Read the industry project material
- Watched webinar 2
- Watched webinar 1
- Gone through all posts in the digital discussion room
- Posted queries and doubts about the project and webinar 1 in DDR
- I went through the linear regression YouTube video
- Read the linear regression article
- Made myself clear with the math behind the model
- Implemented a linear regression model on my own
- Did some participation in the digital discussion room
- Searched and found a proper data set for this project
- Wrote activity reports
- Checked and clarified the data set to determine whether it has enough data for the project
- Read 2 articles and find out how to clean and sanitize the data
- Again, checked and clarified the data set whether it has enough data for the project
- Cleaned the data set
- Sanitized the data set
- Did visualization of the dataset using all the attributes.
- Pre-processed the dataset
- Trained and tested the dataset using logistic regression
- Created classification report

- Did the hyperparameter tuning and generated a classification report using the new model.
- Tested the data against a user-defined data tuple.
- Created a Power BI dashboard on Salary Prediction.
- Went through the following articles and tutorials:

1. https://tirendazacademy.medium.com/machine-learning-project-with-linear-regression-algorithm-b433d770fefd
2. https://www.cloudfactory.com/training-data-guide
3. https://www.w3schools.com/python/python_ml_train_test.asp
4. https://www.geeksforgeeks.org/python-tuples/
5. https://www.geeksforgeeks.org/linear-regression-python-implementation/

## Approach / Methodology

Project methodology refers to the structured approach and systematic processes undertaken to achieve the objectives of a project. It typically involves defining the problem, collecting and preprocessing data, performing exploratory data analysis (EDA), developing and optimizing models, and deploying the solution. In this project, the methodology included sourcing a dataset, cleansing and analyzing the data, applying machine learning algorithms such as Logistic Regression, Random Forest, and SVM, conducting hyperparameter tuning for optimization, validating model performance with metrics, and integrating the final model into a dashboard for real-time salary prediction by HR professionals. This streamlined approach ensures efficient problem-solving and actionable outcomes.

## Outcome

After completing the 1st milestone of this TCS-iON internship project, I have learned how about regression models and I have also successfully cleaned and sanitized the dataset. I have cleaned and removed 2 unnecessary columns capital-gain and capital-loss which are not required in salary prediction.

The following are some of the things I've learned during the last 22 days as a result of my activities.

**Linear Regression**

Linear regression is perhaps one of the most well-known and well-understood algorithms in statistics and machine learning. It is a kind of regression in which we try to fit a line onto a set of data points. It is one of the most common algorithms that's used for predictive analysis. The base of the model is the relation between a dependent and independent variable represented as

$$y = \beta_0 + \beta_1 X + \varepsilon$$

- **y** is the predicted value of the dependent variable (**y**) for any given value of the independent variable (**x**).
- **$B_0$** is the **intercept**, the predicted value of **y** when the **x** is 0.
- **$B_1$** is the regression coefficient – how much we expect **y** to change as **x** increases.
- **x** is the independent variable ( the variable we expect is influencing **y**).
- **e** is the **error** of the estimate, or how much variation there is in our estimate of the regression coefficient.

Linear regression finds the line of best fit line through your data by searching for the regression coefficient ($B_1$) that minimizes the total error (e) of the model.

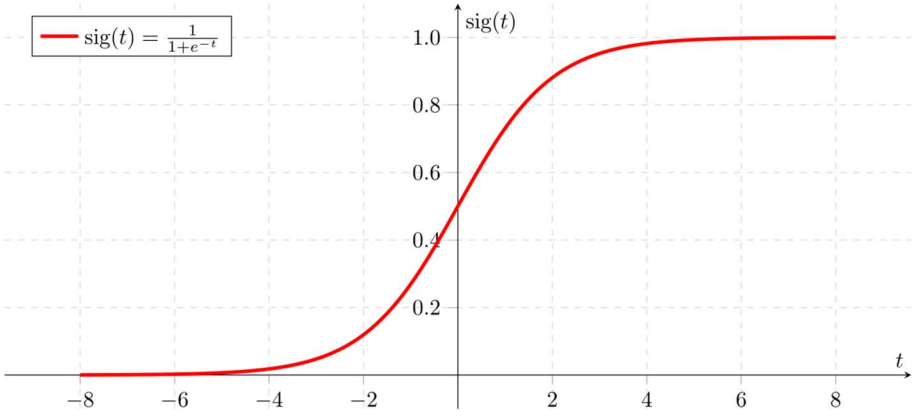
By differentiating the above formula, we can obtain an equation for beta1 and beta2 using which we can define the equation for error. Then we will try to define the model by minimizing the residual error.

$$b_0 = \bar{y} - b_1\bar{x} \qquad b_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

**Logistic Regression**

Logistic Regression is used when the dependent variable(target) is categorical.

The type of function used here is a sigmoid function.




If 'Z' goes to infinity, Y(predicted) will become 1 and if 'Z' goes to negative infinity, Y(predicted) will become 0. So, the only outputs of a logistic regression model are '0' and '1'.

logistic(η) = $\frac{1}{1+\exp(-\eta)}$

The step from linear regression to logistic regression is kind of straightforward. In the linear regression model, we have modelled the relationship between outcome and features with a linear equation:

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}$$

For classification, we prefer probabilities between 0 and 1, so we wrap the right side of the equation into the logistic function. This forces the output to assume only values between 0 and 1.

$$P(y^{(i)} = 1) = \frac{1}{1 + exp(-(\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}))}$$

From the above equation, in the end we can define the odds ratio as

$$\frac{P(y=1)}{1-P(y=1)} = odds = exp\left(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p\right)$$

**Project Development**

Now at the end of 2nd milestone, I have preprocessed the dataset. I have also trained the model using logistic regression and tested the model. Then I have created classification report. Then I have tuned the parameters of the model and chose the best ones. And again, I have tested and printed the classification report. Even though there is not much change in the precision, we can see a small increase in the '<=50k' class. At the end I have tested the model against a user defined data tuple also.

**Project Inference**

So, I have tried a logistic regression model here for the HR salary dashboard. And my classification report was like this.

```
prediction = classifier.predict(x_test)
#generate report
print(classification_report(y_test, prediction))
```

```
              precision    recall  f1-score   support

       <=50K       0.80      0.94      0.86      7455
        >50K       0.55      0.24      0.33      2314

    accuracy                           0.77      9769
   macro avg       0.68      0.59      0.60      9769
weighted avg       0.74      0.77      0.74      9769
```

We can see that the model could achieve a 77% accuracy on predicting the salary. Then I tried it with the SVM model and my classification report was like this

```
from sklearn.svm import SVC
svc = SVC(kernel='linear')
svc.fit(x_train, y_train)
y_pred = svc.predict(x_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       <=50K       0.76      1.00      0.87      7455
        >50K       0.00      0.00      0.00      2314

    accuracy                           0.76      9769
   macro avg       0.38      0.50      0.43      9769
weighted avg       0.58      0.76      0.66      9769
```

I found a decrease in the accuracy, so I continued with my logistic regression model.

After that, I tried it with the Random Forest model and my classification report was like this

```
y_pred = model.predict(x_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       <=50K       0.86      0.90      0.88      7455
        >50K       0.63      0.54      0.58      2314

    accuracy                           0.82      9769
   macro avg       0.75      0.72      0.73      9769
weighted avg       0.81      0.82      0.81      9769
```

We can see that the accuracy achieved by the Random Forest model is 82% in predicting the salary, which is the highest accuracy among all the models.

Now I went on to adjust the parameters of my model even though there was not much biasness in the prediction of 2 classes. So I did the hyperparameter tuning and the results were as follows.

```
Best: 0.980667 using {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}
0.971667 (0.015934) with: {'C': 100, 'penalty': 'l2', 'solver': 'newton-cg'}
0.980000 (0.013166) with: {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
0.970333 (0.016017) with: {'C': 100, 'penalty': 'l2', 'solver': 'liblinear'}
0.971667 (0.015934) with: {'C': 10, 'penalty': 'l2', 'solver': 'newton-cg'}
0.980667 (0.013400) with: {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}
0.970333 (0.016017) with: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
0.971000 (0.014686) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'}
0.972667 (0.016111) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'lbfgs'}
0.970333 (0.015380) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'}
0.971667 (0.015723) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'newton-cg'}
0.973000 (0.015737) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
0.972000 (0.015578) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
0.974667 (0.014772) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'newton-cg'}
0.974000 (0.015832) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
0.974000 (0.015832) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'liblinear'}
```

Hence, I found the best parameters, and then I trained the model using them. The classification report is as follows.

```
grid_predictions = grid.predict(x_test)
# print classification report
print(classification_report(y_test, grid_predictions))
```

```
              precision    recall  f1-score   support

       <=50K       0.85      0.93      0.89      7455
        >50K       0.67      0.49      0.57      2314

    accuracy                           0.82      9769
   macro avg       0.76      0.71      0.73      9769
weighted avg       0.81      0.82      0.81      9769
```

There is a small increase in the precision of '>50k' class.

Then I went onto test my model on a particular user defined tuple of data.

```
Enter the age : 26
Enter the work-class : Private
Enter the education level : Bachelors
Enter the marital-status : Never-married
Enter the occupation : Armed-Forces
Enter the relationship status : Unmarried
Enter the race : White
Enter the sex : Male
Enter the no. of hours he/she work per week : 60
Enter the native-country : United-States
```

The output was as following.

```
The salary will be <=50K
```

## Link to code and executable file

Link to the colab file:

https://colab.research.google.com/drive/1b3j-E2XSyrCbPhyy6aM3A4YlXxUowgoM

Link to the GitHub file:

https://github.com/Archita05D/TCS-ION-RIO-210-Salary-Prediction-Dashboard