Course Topic: Linear Regression

Data Used:

Jobstreet.csv

Source of Data:

https://drive.google.com/file/d/1Aj-QZY0dRH4PxA_QzzlWOVSGSrVEJacd/view?usp=drive_link
https://github.com/PioBurgosGithub/Jobstreet_LinearRegression

Data Information:

The dataset used in this analysis includes information on job positions, years of experience, and salary rates. Job positions represent positions undertaken by individuals to perform work within the dataset. Years of experience indicate the duration of professional experience typically required for each job position. Salary rates correspond to the monetary compensation provided to employees for their work. The dataset provides a comprehensive representation of these variables, allowing for an exploration of the relationship between years of experience and salary rates across different job positions.

Problem Statement:

The task of this activity is to scrape job data from Indeed, build a linear regression model to estimate salary rates based on experience, assess the model's accuracy and variance, and deploy the model using Pickle for future utilization.

Method:

Algorithm:

Firstly, we scrape job data from Jobstreet.com, extracting job titles, required experience, and salary rates. The scraped data is stored in a list, and we ensure to scrape a specific number of job listings within a given range. Next, we utilize linear regression analysis to estimate salary rates based on job experience. The scraped data is divided into input features and target variables, and a linear regression model is trained using this data. We evaluate the accuracy of the model and analyze its performance using variance analysis. Finally, we deploy the trained model using the Pickle package.

Solution:

The solution consists of three main components: data scraping, linear regression analysis, and model deployment. Firstly, data scraping involves using web scraping techniques like BeautifulSoup or Scrapy to extract job listings from Jobstreet.com, job titles, required

experience, and salary rates. The scraped data is then organized and stored in a structured format, such as a list of dictionaries. Moving on to the linear regression analysis, the scraped data is divided into input features (job experience) and the target variable (salary rate). With the help of libraries like sci-kit-learn, a linear regression model is constructed and trained using the input features and target variable. The model is then used to make predictions on salary rates based on job experience. The trained linear regression model is deployed using the Pickle package in the final step. The model is serialized and saved to a file, enabling its future use. When required, the model can be loaded from the file using the Pickle package and utilized to make predictions on new data by providing the appropriate input features.

Evaluation:

This activity involves carefully assessing each component of the solution to determine its effectiveness and performance. This includes evaluating the data scraping phase to ensure the accuracy and completeness of the scraped data, as well as verifying its consistency and alignment with the original Indeed pages. The linear regression model is evaluated by analyzing its predictions against the actual values from the testing dataset, using appropriate evaluation metrics. The model deployment process is assessed by verifying the successful installation of the Pickle package, checking the integrity of the saved model file, and testing the reloaded model's accuracy in making predictions on new data.

Data Preparation:

In this activity, these are the following processes we have done in data preparation.

1. **Extract Data – in this activity we've extracted data from the website**
   https://www.jobstreet.com.ph/en/job-search/junior-developer-jobs/
   https://www.jobstreet.com.ph/en/job-search/senior-developer-jobs/
   https://www.jobstreet.com.ph/project-manager-jobs
   https://www.jobstreet.com.ph/cto-jobs

2. Scrape the required data from the Indeed pages, including job titles (Junior, Senior, CTO, PM), required experience, and salary rates.
   2.1. Ensure that the scraping process retrieves a minimum of 100 and a maximum of 150 job listings to have a sufficient amount of data for analysis.

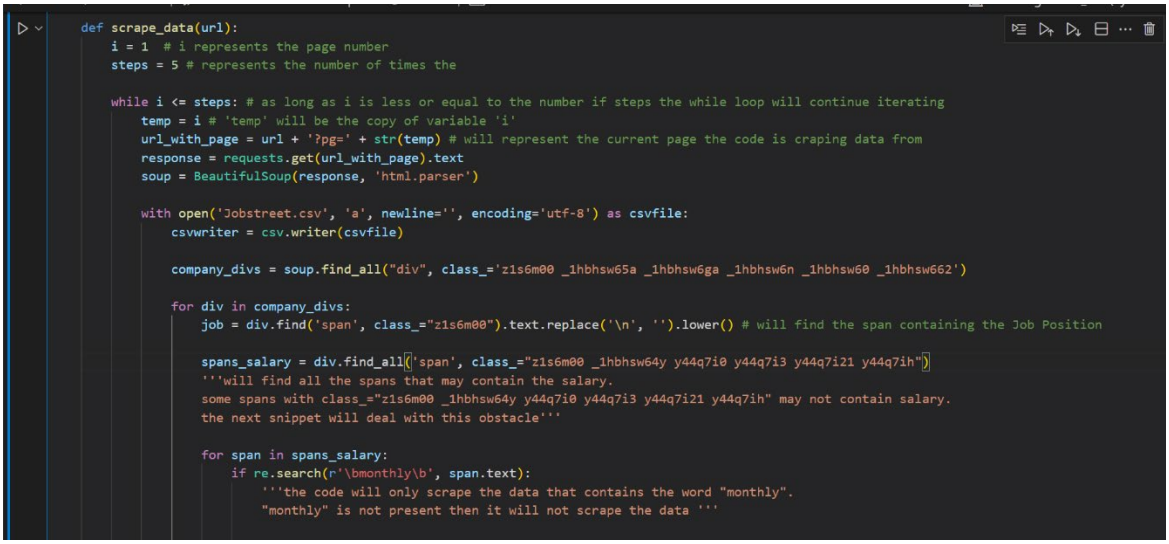
3. Split the scraped data into input features and the target variable. In this case, the job experience will be the input feature, and the salary rate will be the target variable.
   3.1. Apply linear regression to estimate the salary rate based on the job experience.

4. Install the Pickle package using pip to enable serialization and deserialization of the model.

4.1 Dump the trained linear regression model using Pickle, which serializes the model and saves it to a file.

Results & Discussion

Data Scraping



```
def scrape_data(url):
    i = 1  # i represents the page number
    steps = 5 # represents the number of times the

    while i <= steps: # as long as i is less or equal to the number if steps the while loop will continue iterating
        temp = i # 'temp' will be the copy of variable 'i'
        url_with_page = url + '?pg=' + str(temp) # will represent the current page the code is craping data from
        response = requests.get(url_with_page).text
        soup = BeautifulSoup(response, 'html.parser')

        with open('Jobstreet.csv', 'a', newline='', encoding='utf-8') as csvfile:
            csvwriter = csv.writer(csvfile)

            company_divs = soup.find_all("div", class_='z1s6m00 _1hbhsw65a _1hbhsw6ga _1hbhsw6n _1hbhsw60 _1hbhsw662')

            for div in company_divs:
                job = div.find('span', class_="z1s6m00").text.replace('\n', '').lower() # will find the span containing the Job Position

                spans_salary = div.find_all('span', class_="z1s6m00 _1hbhsw64y y44q7i0 y44q7i3 y44q7i21 y44q7ih")
                '''will find all the spans that may contain the salary.
                some spans with class_="z1s6m00 _1hbhsw64y y44q7i0 y44q7i3 y44q7i21 y44q7ih" may not contain salary.
                the next snippet will deal with this obstacle'''

                for span in spans_salary:
                    if re.search(r'\bmonthly\b', span.text):
                        '''the code will only scrape the data that contains the word "monthly".
                        "monthly" is not present then it will not scrape the data '''
```

```
                salary_str = re.findall(r'\S+\s*\S+', span.text)[0]
                salary_str = re.sub(r'[^\d\.]', '', salary_str)
                salary = float(salary_str) * 1000
                #This code snippet will convert salaries like Php15k or Php50k strings into 15000.0 or 50000.0 floats

                link = div.find("a")
                link = link['href']
                # these lines code will be used to find the Relative URL present in the div

                base_url = "https://www.jobstreet.com.ph"
                link = urllib.parse.urljoin(base_url, link) # this code is used to join the base_url with the relative

                hyperlink_response = requests.get(link, timeout=5)
                hyperlink_soup = BeautifulSoup(hyperlink_response.content, "html.parser")
                '''this code snippet will make a request to the URL link using the new Base_Url.
                timeout argument specifies the amount of time to wait for the request to complete.
                If the request does not complete within the specified time, the request will be aborted
                then BeautifulSoup will extract data from the HTML pages.'''


                #'''After the request has been sent and rthe code will scrape the data in the new hyperlink'''
                experiences_spans = hyperlink_soup.find_all('span', class_="z1s6m00 _1hbhsw64y y44q7i0 y44q7i1 y44q7i21 _1d0g9qk4 y44q7ia")
                for x in experiences_spans:
                    if re.search(r'\byears\b', x.text):
                        ''''the code will only scrape the data that contains the word "years".
                        "years" is not present then it will not scrape the data '''
                        experience = re.findall(r'\S+\s*\S+', x.text)[0]
```

```
                salary_str = re.sub(r'[^\d\.]', '', salary_str)
                salary = float(salary_str) * 1000
                #This code snippet will convert salaries like Php15k or Php50k strings into 15000.0 or 50000.0 floats

                link = div.find("a")
                link = link['href']
                # these lines code will be used to find the Relative URL present in the div

                base_url = "https://www.jobstreet.com.ph"
                link = urllib.parse.urljoin(base_url, link) # this code is used to join the base_url with the relative

                hyperlink_response = requests.get(link, timeout=5)
                hyperlink_soup = BeautifulSoup(hyperlink_response.content, "html.parser")
                '''this code snippet will make a request to the URL link using the new Base_Url.
                timeout argument specifies the amount of time to wait for the request to complete.
                If the request does not complete within the specified time, the request will be aborted
                then BeautifulSoup will extract data from the HTML pages.'''


                #'''After the request has been sent and rthe code will scrape the data in the new hyperlink'''
                experiences_spans = hyperlink_soup.find_all('span', class_="z1s6m00 _1hbhsw64y y44q7i0 y44q7i1 y44q7i21 _1d0g9qk4 y44q7ia")
                for x in experiences_spans:
                    if re.search(r'\byears\b', x.text):
                        ''''the code will only scrape the data that contains the word "years".
                        "years" is not present then it will not scrape the data '''
                        experience = re.findall(r'\S+\s*\S+', x.text)[0]
                        experience = re.sub(r'[^\d\.]', '', experience)

                        csvwriter.writerow([job, experience, link, salary]) # this line of code will write all the data scraped into 'Jobstreet.csv'

    print("Page", i, "& step", steps, "\n")
    i += 1
```

✓ 0.1s

Shows the dataset we will use. We shall make a copy of the dataset so that the original will not be altered

```
# this will show the dataset used for linear regression
dataset = pd.read_csv('Jobstreet.csv')
dataset
```

[71] ✓ 0.2s

	Job	Experience	URLs	Salary
0	junior programmer	2.0	https://www.jobstreet.com.ph/en/job/junior-pro...	15000.0
1	programmer/software engineer	2.0	https://www.jobstreet.com.ph/en/job/programmer...	25000.0
2	junior java developer	2.0	https://www.jobstreet.com.ph/en/job/junior-jav...	25000.0
3	junior backend developer	2.0	https://www.jobstreet.com.ph/en/job/junior-bac...	20000.0
4	android developer (junior, mid, senior)	3.0	https://www.jobstreet.com.ph/en/job/android-de...	45000.0
...	...	...	...	...
110	systems administrator (work from home)	5.0	https://www.jobstreet.com.ph/en/job/systems-ad...	50000.0
111	software engineer - python centric	3.0	https://www.jobstreet.com.ph/en/job/software-e...	100000.0
112	property accountant | hybrid, night-shift | up...	5.0	https://www.jobstreet.com.ph/en/job/property-a...	45000.0
113	ao ii-fad hr -28-23	4.0	https://www.jobstreet.com.ph/en/job/ao-ii-fad-...	32000.0
114	construction project coordinator | wfh (cebu),...	4.0	https://www.jobstreet.com.ph/en/job/constructi...	45000.0

115 rows × 4 columns

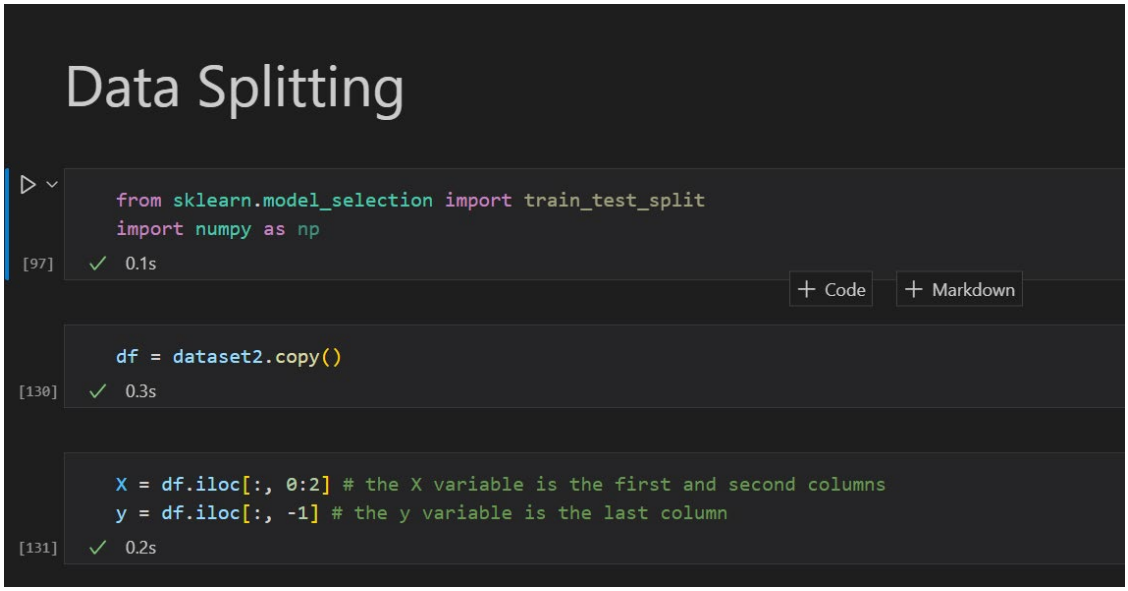
```
dataset2 = dataset.copy() # to make a copy of the dataset. We can alter the copied dataset as many times we want

dataset2['Job'] = dataset2['Job'].astype(str)
print(type(dataset2['Job']))
```

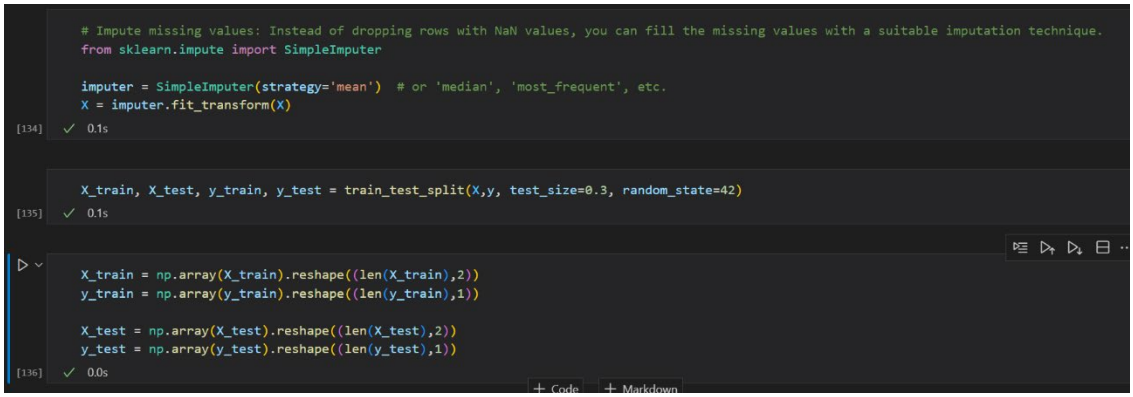
[87] ✓ 0.5s

```
# Define the regular expressions to match the job titles

junior_regex = r'\bjunior\b'
senior_regex = r'\bsenior developer\b|\bdeveloper\b|\bengineer\b|\bprogrammer\b'
pm_regex = r'\bproject manager\b'

''' Define a lambda function to replace the job titles with 0, 1, 2, or 3
0 == Junior Developer
1 == Senior Developer
2 == Project Manager
3 == titles that don't match the previous examples'''
replace_func = lambda x: 0 if re.search(junior_regex, x) else 1 if re.search(senior_regex, x) else 2 if re.search(pm_regex, x) else 3

# Apply the lambda function to the Job column and assign the result to a new column
dataset2['Job'] = dataset2['Job'].apply(replace_func)
```

[88] ✓ 2.8s Python

Data Splitting. This code locates which columns are the independent and dependent variables that will be used for the linear regression model





Model Training

```
[104]  from sklearn import linear_model
```
✓ 0.0s

```
[137]  model = linear_model.LinearRegression()
       model.fit(X_train,y_train)
       #this snippet of code will be used to train the linearRegression model using the X and y variables.
```
✓ 0.2s

```
LinearRegression
LinearRegression()
```

```
[138]  model.intercept_
```
✓ 0.1s

```
array([40564.58265946])
```

```
[139]  model.coef_
```
✓ 0.0s

```
array([[-2915.3046141 ,  8016.71836712]])
```

```
[140]  model.score(X_train,y_train)
```
✓ 0.1s

```
0.2401265211971758
```

```
[141]  y_pred = model.predict(X_test)
```
✓ 0.0s

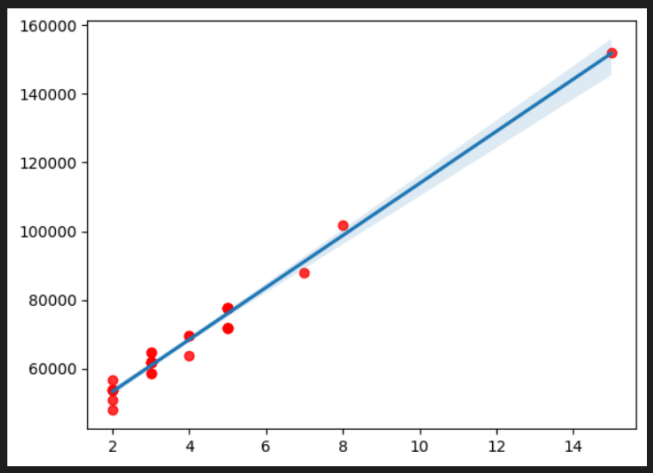
```
[142]  y_pred
```
✓ 0.0s

```
array([[ 58784.12853263],
       [ 64614.73776084],
       [ 61699.43314673],
       [ 77732.86988098],
       [ 53682.71477961],
       [ 77732.86988098],
       [ 77732.86988098],
       [ 77732.86988098],
       [ 61699.43314673],
       [ 64614.73776084],
       [152069.44432401],
       [ 77732.86988098],
       [ 50767.41016551],
       [ 63885.54228565],
       [ 61699.43314673],
       [ 56598.01939371],
       [ 77732.86988098],
       [ 53682.71477961],
       [ 47852.1055514 ],
       [ 69716.15151386],
       [ 71902.26065277],
       [ 71902.26065277],
       [ 61699.43314673],
       [ 61699.43314673],
```

```
plt.figure(10.5)
sns.regplot(x=X_test[:,1], y = y_pred, scatter_kws={'color':'red'})
```

[143] ✓ 4.9s

```
<Axes: >
```




```
plt.show()
```

[125] ✓ 0.1s

Save Model

```
import pickle
```

[126] ✓ 0.0s

```
pickle.dump(model, open('model.pkl','wb'))
```

[127] ✓ 0.1s

```
model_dump = pickle.load(open('model.pkl','rb'))
```

[128] ✓ 0.1s

```
print(model.predict([[1.2,100]]))
```

[129] ✓ 0.1s

```
[[838738.05383494]]
```

The linear regression analysis conducted explored the relationship between years of experience and salary. By employing the principles of linear regression, this analysis sought to uncover any potential patterns or trends in the data that shed light on how the experience influenced salary levels. By examining the relationship between these two variables, we gained insights into the extent to which an individual's accumulated years of experience impacted their earning potential. The analysis determined how years of experience affect salary in job positions. Based on the results specifically the results of the given data shown in the scatterplot which serves as an interpretation of the relationship between the values. The data points are spread out around the regression line, indicating variability in predicted salaries at each experience level. This suggests that factors other than experience, such as education, industry, or job position, influence salaries. While the regression line shows a positive trend, indicating increasing predicted salaries with higher experience, the scatter plot highlights deviations between the values.