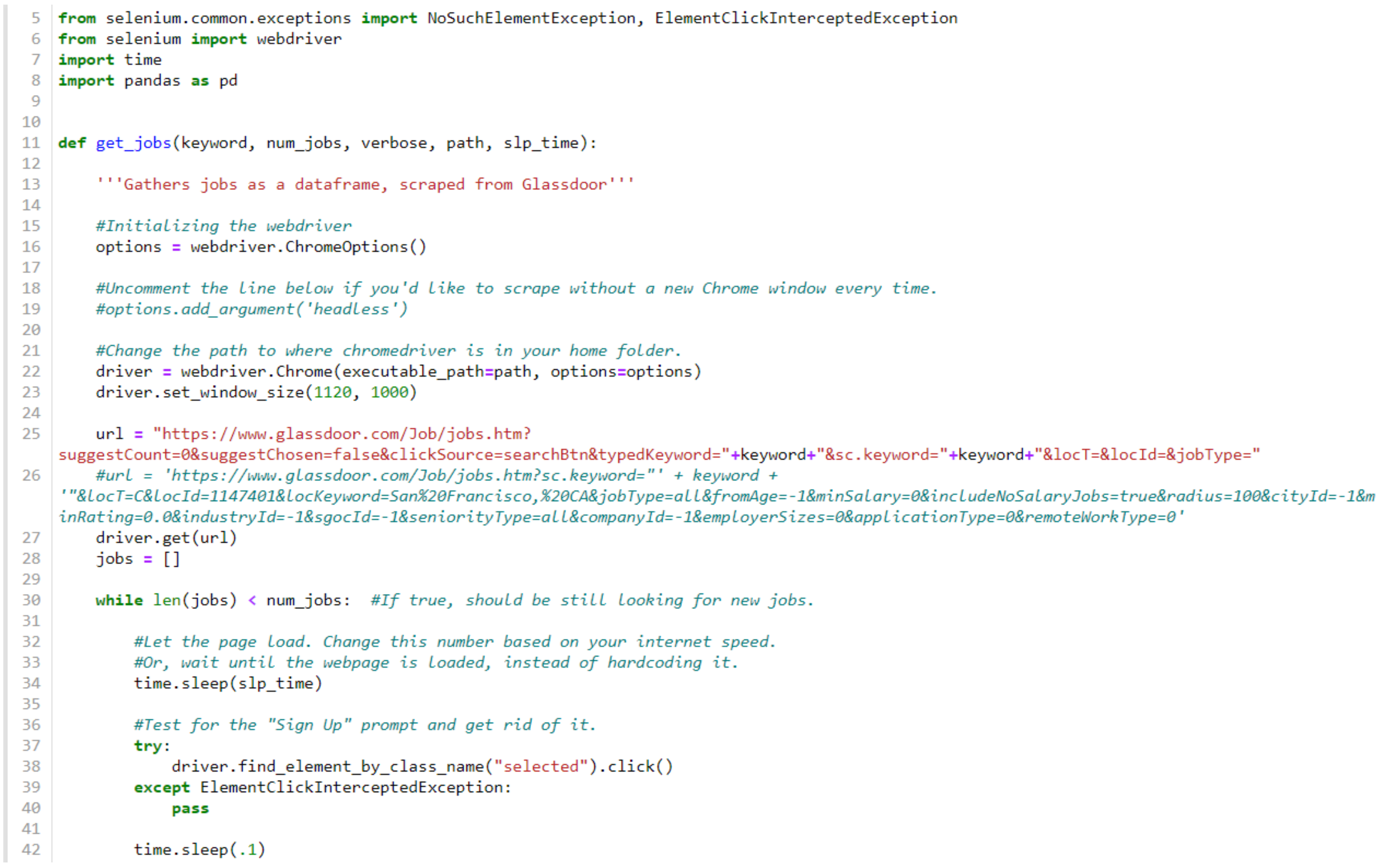
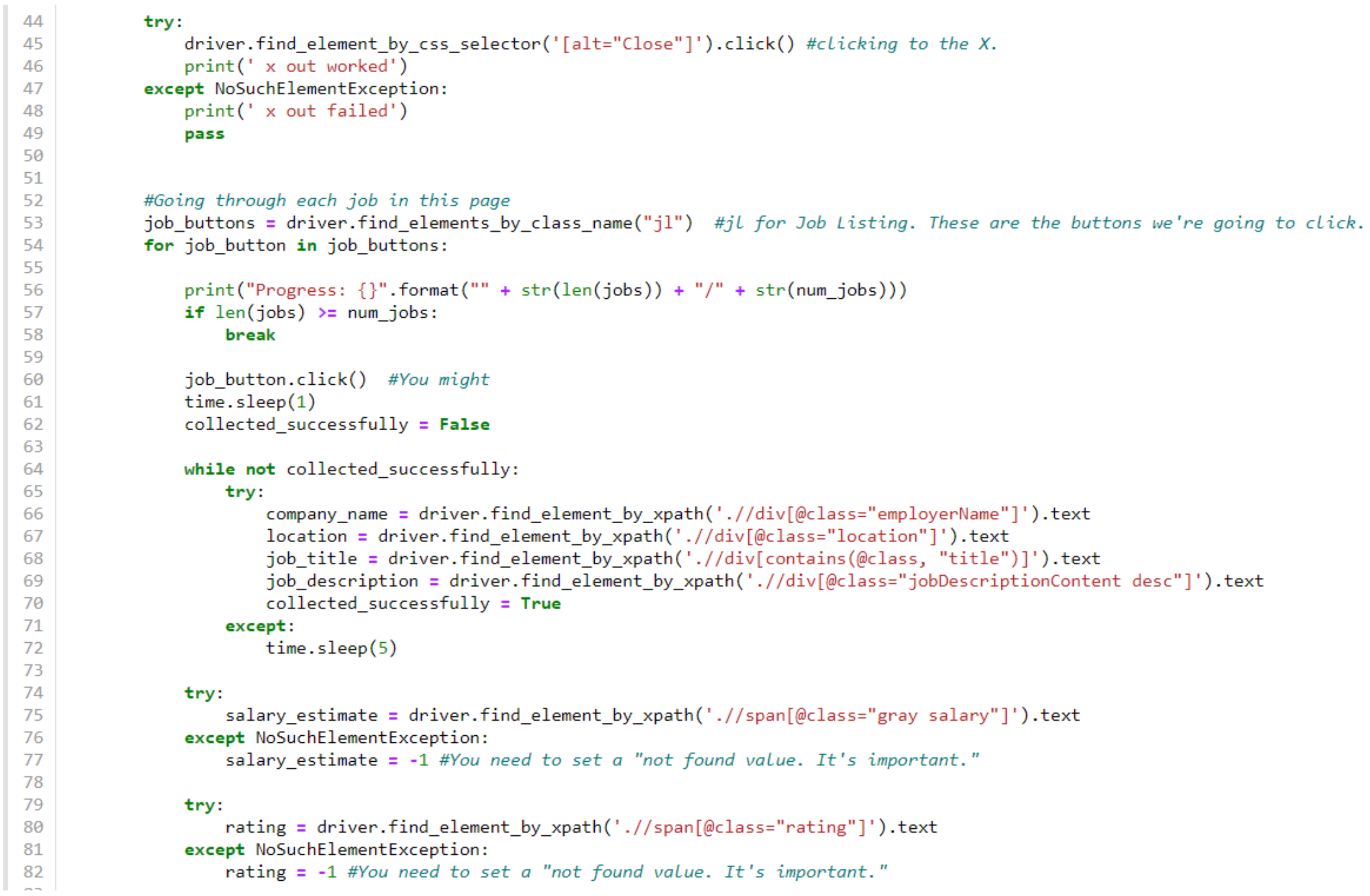
# Project Name: End-To-End ML With Deployment Salary Prediction – Linear – Regression

## Table of Contents

Demo



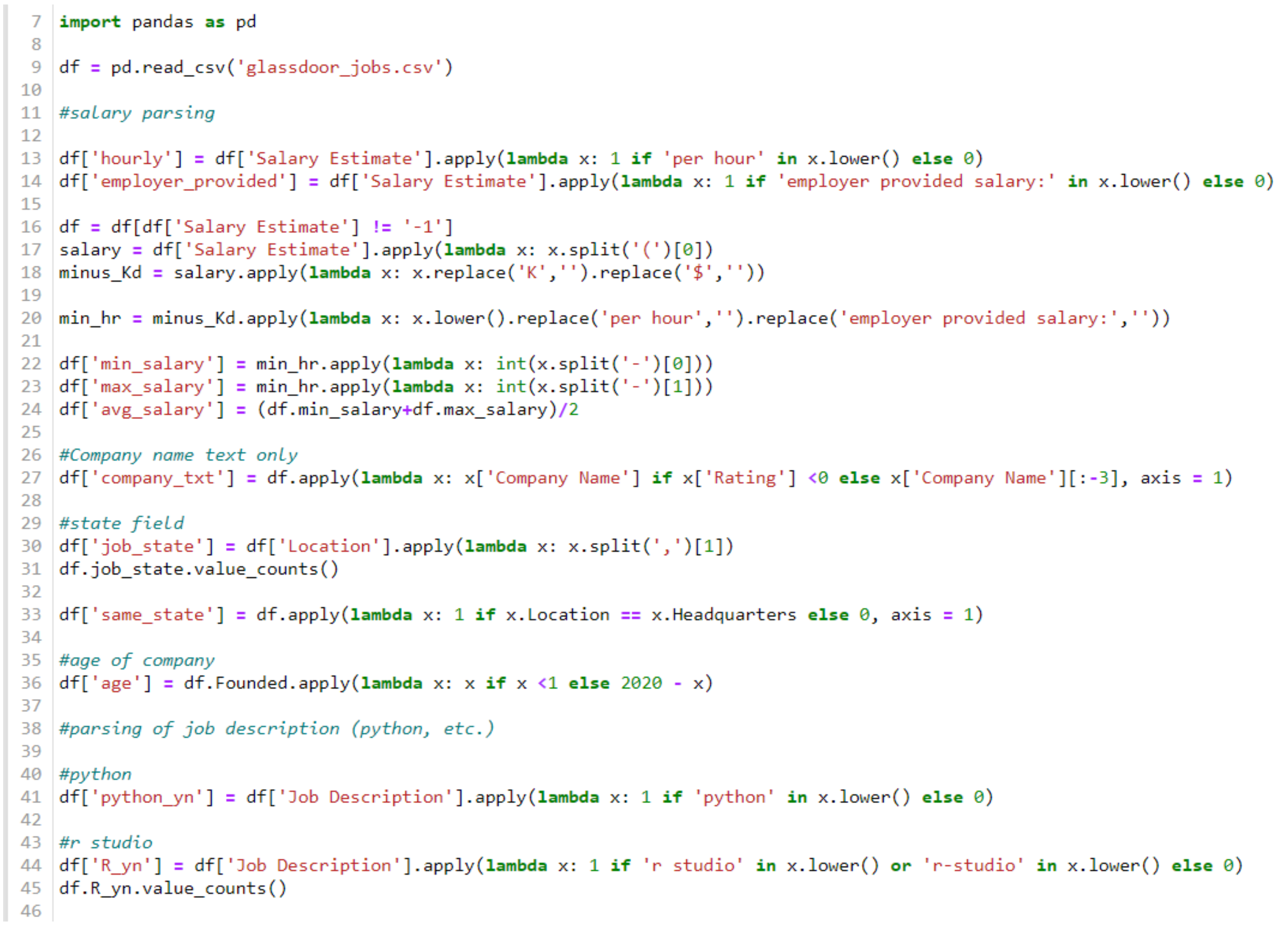
```python
from selenium.common.exceptions import NoSuchElementException, ElementClickInterceptedException
from selenium import webdriver
import time
import pandas as pd


def get_jobs(keyword, num_jobs, verbose, path, slp_time):

    '''Gathers jobs as a dataframe, scraped from Glassdoor'''

    #Initializing the webdriver
    options = webdriver.ChromeOptions()

    #Uncomment the line below if you'd like to scrape without a new Chrome window every time.
    #options.add_argument('headless')

    #Change the path to where chromedriver is in your home folder.
    driver = webdriver.Chrome(executable_path=path, options=options)
    driver.set_window_size(1120, 1000)

    url = "https://www.glassdoor.com/Job/jobs.htm?suggestCount=0&suggestChosen=false&clickSource=searchBtn&typedKeyword="+keyword+"&sc.keyword="+keyword+"&locT=&locId=&jobType="
    #url = 'https://www.glassdoor.com/Job/jobs.htm?sc.keyword="' + keyword + '"&locT=C&locId=1147401&locKeyword=San%20Francisco,%20CA&jobType=all&fromAge=-1&minSalary=0&includeNoSalaryJobs=true&radius=100&cityId=-1&minRating=0.0&industryId=-1&sgocId=-1&seniorityType=all&companyId=-1&employerSizes=0&applicationType=0&remoteWorkType=0'
    driver.get(url)
    jobs = []

    while len(jobs) < num_jobs:  #If true, should be still looking for new jobs.

        #Let the page load. Change this number based on your internet speed.
        #Or, wait until the webpage is loaded, instead of hardcoding it.
        time.sleep(slp_time)

        #Test for the "Sign Up" prompt and get rid of it.
        try:
            driver.find_element_by_class_name("selected").click()
        except ElementClickInterceptedException:
            pass

        time.sleep(.1)
```



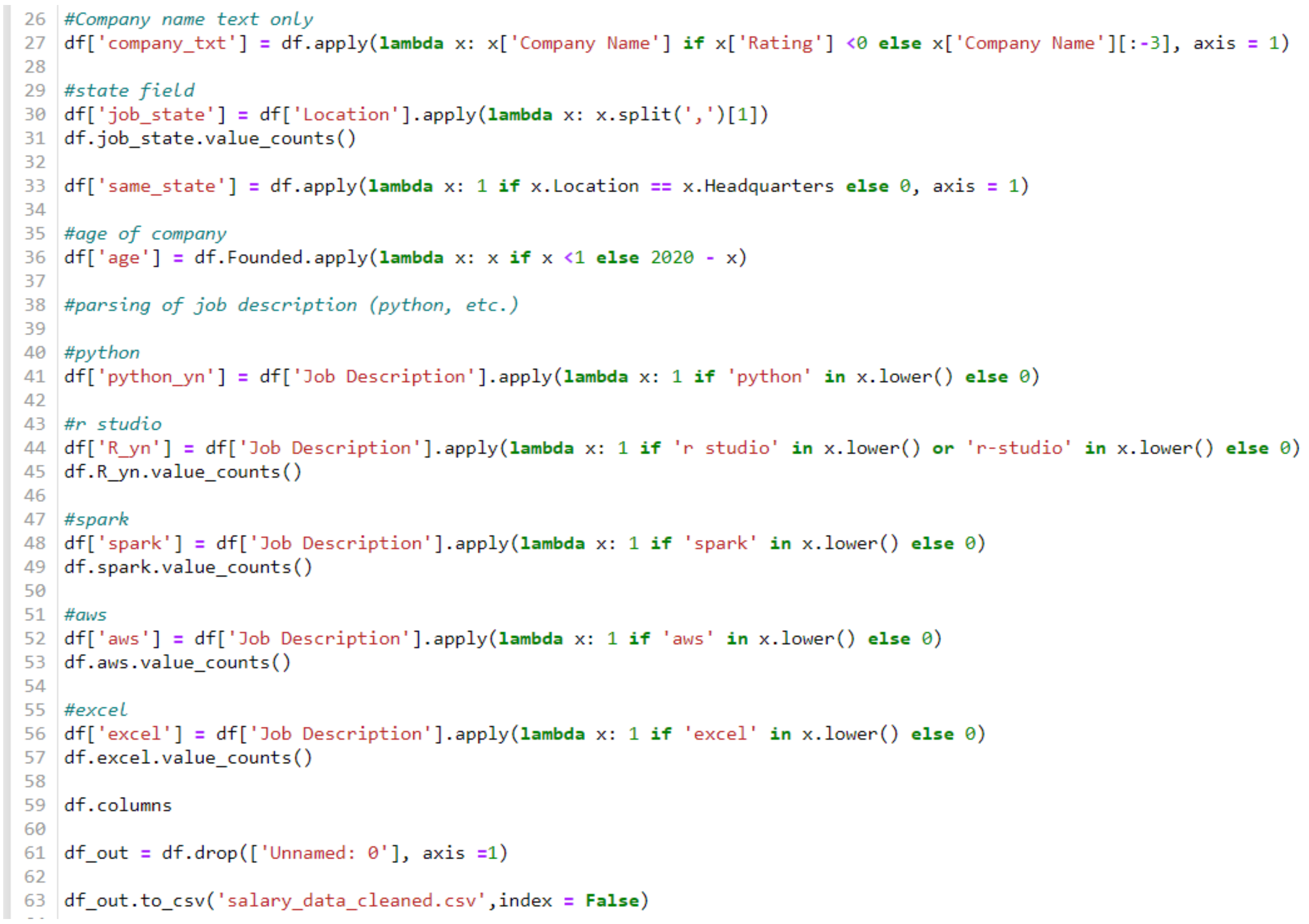
```python
        try:
            driver.find_element_by_css_selector('[alt="Close"]').click() #clicking to the X.
            print(' x out worked')
        except NoSuchElementException:
            print(' x out failed')
            pass


        #Going through each job in this page
        job_buttons = driver.find_elements_by_class_name("jl")  #jl for Job Listing. These are the buttons we're going to click.
        for job_button in job_buttons:

            print("Progress: {}".format("" + str(len(jobs)) + "/" + str(num_jobs)))
            if len(jobs) >= num_jobs:
                break

            job_button.click()  #You might
            time.sleep(1)
            collected_successfully = False

            while not collected_successfully:
                try:
                    company_name = driver.find_element_by_xpath('.//div[@class="employerName"]').text
                    location = driver.find_element_by_xpath('.//div[@class="location"]').text
                    job_title = driver.find_element_by_xpath('.//div[contains(@class, "title")]').text
                    job_description = driver.find_element_by_xpath('.//div[@class="jobDescriptionContent desc"]').text
                    collected_successfully = True
                except:
                    time.sleep(5)

            try:
                salary_estimate = driver.find_element_by_xpath('.//span[@class="gray salary"]').text
            except NoSuchElementException:
                salary_estimate = -1 #You need to set a "not found value. It's important."

            try:
                rating = driver.find_element_by_xpath('.//span[@class="rating"]').text
            except NoSuchElementException:
                rating = -1 #You need to set a "not found value. It's important."
            #Printing for debugging
            if verbose:
                print("Job Title: {}".format(job_title))
                print("Salary Estimate: {}".format(salary_estimate))
                print("Job Description: {}".format(job_description[:500]))
                print("Rating: {}".format(rating))
                print("Company Name: {}".format(company_name))
                print("Location: {}".format(location))

            #Going to the Company tab...
            #clicking on this:
            #<div class="tab" data-tab-type="overview"><span>Company</span></div>
            try:
                driver.find_element_by_xpath('.//div[@class="tab" and @data-tab-type="overview"]').click()

                try:
                    #<div class="infoEntity">
                    #    <label>Headquarters</label>
                    #    <span class="value">San Francisco, CA</span>
                    #</div>
                    headquarters = driver.find_element_by_xpath('.//div[@class="infoEntity"]//label[text()="Headquarters"]//following-sibling::*').text
                except NoSuchElementException:
                    headquarters = -1

                try:
                    size = driver.find_element_by_xpath('.//div[@class="infoEntity"]//label[text()="Size"]//following-sibling::*').text
                except NoSuchElementException:
                    size = -1

                try:
                    founded = driver.find_element_by_xpath('.//div[@class="infoEntity"]//label[text()="Founded"]//following-sibling::*').text
                except NoSuchElementException:
                    founded = -1

                try:
                    type_of_ownership = driver.find_element_by_xpath('.//div[@class="infoEntity"]//label[text()="Type"]//following-sibling::*').text
                except NoSuchElementException:
                    type_of_ownership = -1
```

```python
                try:
                    industry = driver.find_element_by_xpath('.//div[@class="infoEntity"]//label[text()="Industry"]//following-sibling::*').text
                except NoSuchElementException:
                    industry = -1

                try:
                    sector = driver.find_element_by_xpath('.//div[@class="infoEntity"]//label[text()="Sector"]//following-sibling::*').text
                except NoSuchElementException:
                    sector = -1

                try:
                    revenue = driver.find_element_by_xpath('.//div[@class="infoEntity"]//label[text()="Revenue"]//following-sibling::*').text
                except NoSuchElementException:
                    revenue = -1

                try:
                    competitors = driver.find_element_by_xpath('.//div[@class="infoEntity"]//label[text()="Competitors"]//following-sibling::*').text
                except NoSuchElementException:
                    competitors = -1

            except NoSuchElementException:  #Rarely, some job postings do not have the "Company" tab.
                headquarters = -1
                size = -1
                founded = -1
                type_of_ownership = -1
                industry = -1
                sector = -1
                revenue = -1
                competitors = -1
            if verbose:
                print("Headquarters: {}".format(headquarters))
                print("Size: {}".format(size))
                print("Founded: {}".format(founded))
                print("Type of Ownership: {}".format(type_of_ownership))
                print("Industry: {}".format(industry))
                print("Sector: {}".format(sector))
                print("Revenue: {}".format(revenue))
                print("Competitors: {}".format(competitors))
                print("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@")

            jobs.append({"Job Title" : job_title,
            "Salary Estimate" : salary_estimate,
            "Job Description" : job_description,
            "Rating" : rating,
            "Company Name" : company_name,
            "Location" : location,
            "Headquarters" : headquarters,
            "Size" : size,
            "Founded" : founded,
            "Type of ownership" : type_of_ownership,
            "Industry" : industry,
            "Sector" : sector,
            "Revenue" : revenue,
            "Competitors" : competitors})
            #add job to jobs


        #Clicking on the "next page" button
        try:
            driver.find_element_by_xpath('.//li[@class="next"]//a').click()
        except NoSuchElementException:
            print("Scraping terminated before reaching target number of jobs. Needed {}, got {}.".format(num_jobs, len(jobs)))
            break

    return pd.DataFrame(jobs)  #This line converts the dictionary object into a pandas DataFrame.
```

```python
import glassdoor_scraper as gs
import pandas as pd

#path = "C:/Users/Documents/ds_salary_proj/chromedriver"

#path = "C:/Users/Monica/Desktop/Projects/Python Projects 1/allin1/Project_3_WebScraping+DataCollection+DataCleaning+EDAwithVisualization+ModelBuilding(End_to_End_Project)/chromedriver"

path = "C:/Users/Monica/Desktop/salary_prediction_app/chromedriver.exe"

df = gs.get_jobs('data scientist',1000, False, path, 15)

df.to_csv('glassdoor_jobs.csv', index = False)
```



```python
import pandas as pd

df = pd.read_csv('glassdoor_jobs.csv')

#salary parsing

df['hourly'] = df['Salary Estimate'].apply(lambda x: 1 if 'per hour' in x.lower() else 0)
df['employer_provided'] = df['Salary Estimate'].apply(lambda x: 1 if 'employer provided salary:' in x.lower() else 0)

df = df[df['Salary Estimate'] != '-1']
salary = df['Salary Estimate'].apply(lambda x: x.split('(')[0])
minus_Kd = salary.apply(lambda x: x.replace('K','').replace('$',''))

min_hr = minus_Kd.apply(lambda x: x.lower().replace('per hour','').replace('employer provided salary:',''))

df['min_salary'] = min_hr.apply(lambda x: int(x.split('-')[0]))
df['max_salary'] = min_hr.apply(lambda x: int(x.split('-')[1]))
df['avg_salary'] = (df.min_salary+df.max_salary)/2

#Company name text only
df['company_txt'] = df.apply(lambda x: x['Company Name'] if x['Rating'] <0 else x['Company Name'][:-3], axis = 1)

#state field
df['job_state'] = df['Location'].apply(lambda x: x.split(',')[1])
df.job_state.value_counts()

df['same_state'] = df.apply(lambda x: 1 if x.Location == x.Headquarters else 0, axis = 1)

#age of company
df['age'] = df.Founded.apply(lambda x: x if x <1 else 2020 - x)

#parsing of job description (python, etc.)

#python
df['python_yn'] = df['Job Description'].apply(lambda x: 1 if 'python' in x.lower() else 0)

#r studio
df['R_yn'] = df['Job Description'].apply(lambda x: 1 if 'r studio' in x.lower() or 'r-studio' in x.lower() else 0)
df.R_yn.value_counts()

```



```python
#Company name text only
df['company_txt'] = df.apply(lambda x: x['Company Name'] if x['Rating'] <0 else x['Company Name'][:-3], axis = 1)

#state field
df['job_state'] = df['Location'].apply(lambda x: x.split(',')[1])
df.job_state.value_counts()

df['same_state'] = df.apply(lambda x: 1 if x.Location == x.Headquarters else 0, axis = 1)

#age of company
df['age'] = df.Founded.apply(lambda x: x if x <1 else 2020 - x)

#parsing of job description (python, etc.)

#python
df['python_yn'] = df['Job Description'].apply(lambda x: 1 if 'python' in x.lower() else 0)

#r studio
df['R_yn'] = df['Job Description'].apply(lambda x: 1 if 'r studio' in x.lower() or 'r-studio' in x.lower() else 0)
df.R_yn.value_counts()

```

```python
#spark
df['spark'] = df['Job Description'].apply(lambda x: 1 if 'spark' in x.lower() else 0)
df.spark.value_counts()

#aws
df['aws'] = df['Job Description'].apply(lambda x: 1 if 'aws' in x.lower() else 0)
df.aws.value_counts()

#excel
df['excel'] = df['Job Description'].apply(lambda x: 1 if 'excel' in x.lower() else 0)
df.excel.value_counts()

df.columns

df_out = df.drop(['Unnamed: 0'], axis =1)

df_out.to_csv('salary_data_cleaned.csv',index = False)
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv('salary_data_cleaned.csv')
```

```
df.head()
```

| | Job Title | Salary Estimate | Job Description | Rating | Company Name | Location | Headquarters | Size | Founded | Type of ownership | ... | avg_salary | company_txt |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Data Scientist | $53K-$91K (Glassdoor est.) | Data Scientist\nLocation: Albuquerque, NM\nEdu... | 3.8 | Tecolote Research\n3.8 | Albuquerque, NM | Goleta, CA | 501 to 1000 employees | 1973 | Company - Private | ... | 72.0 | Tecolote Research\n |
| 1 | Healthcare Data Scientist | $63K-$112K (Glassdoor est.) | What You Will Do:\n\nI. General Summary\n\nThe... | 3.4 | University of Maryland Medical System\n3.4 | Linthicum, MD | Baltimore, MD | 10000+ employees | 1984 | Other Organization | ... | 87.5 | University of Maryland Medical System\n |
| 2 | Data Scientist | $80K-$90K (Glassdoor est.) | KnowBe4, Inc. is a high growth information sec... | 4.8 | KnowBe4\n4.8 | Clearwater, FL | Clearwater, FL | 501 to 1000 employees | 2010 | Company - Private | ... | 85.0 | KnowBe4\n |
| 3 | Data Scientist | $56K-$97K (Glassdoor est.) | *Organization and Job ID**\nJob ID: 310709\n\n... | 3.8 | PNNL\n3.8 | Richland, WA | Richland, WA | 1001 to 5000 employees | 1965 | Government | ... | 76.5 | PNNL\n |
| 4 | Data Scientist | $86K-$143K (Glassdoor est.) | Data Scientist\nAffinity Solutions / Marketing... | 2.9 | Affinity Solutions\n2.9 | New York, NY | New York, NY | 51 to 200 employees | 1998 | Company - Private | ... | 114.5 | Affinity Solutions\n |

5 rows × 28 columns

```
df.job_simp.value_counts()
```

```
data scientist    279
na                184
data engineer     119
analyst           102
manager            22
mle                22
director           14
Name: job_simp, dtype: int64
```

```
df['seniority'] = df['Job Title'].apply(seniority)
df.seniority.value_counts()
```

```
na        520
senior    220
jr          2
Name: seniority, dtype: int64
```

```
#Competitor count
df['num_comp'] = df['Competitors'].apply(lambda x: len(x.split(',')) if x != '-1' else 0)
```

```
df['Competitors']
```

```
0                                                     -1
1                                                     -1
2                                                     -1
3      Oak Ridge National Laboratory, National Renewa...
4                  Commerce Signals, Cardlytics, Yodlee
                             ...
737                         Pfizer, AstraZeneca, Merck
738                     See Tickets, TicketWeb, Vendini
739                                                   -1
740                                                   -1
741                                                   -1
Name: Competitors, Length: 742, dtype: object
```

```
df['company_txt'] = df.company_txt.apply(lambda x: x.replace('\n', ''))
```

```
df['company_txt']
```

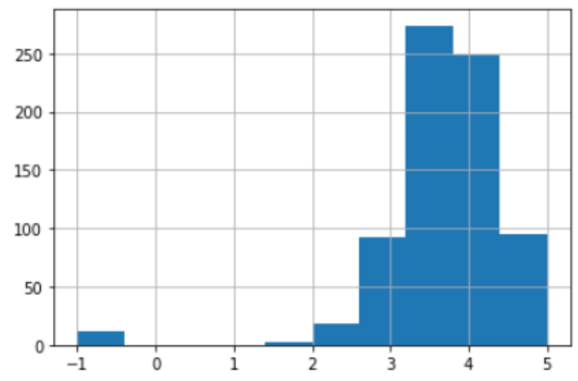
```
0                      Tecolote Research
1       University of Maryland Medical System
2                                KnowBe4
3                                   PNNL
4                     Affinity Solutions
                     ...
737                                  GSK
738                           Eventbrite
739             Software Engineering Institute
740                          Numeric, LLC
741               Riverside Research Institute
Name: company_txt, Length: 742, dtype: object
```

```
df.describe()
```

| | Rating | Founded | hourly | employer_provided | min_salary | max_salary | avg_salary | same_state | age | python_yn | R_yn | spa |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.0000 |
| mean | 3.618868 | 1837.154987 | 0.032345 | 0.022911 | 74.719677 | 128.149596 | 100.626011 | 0.557951 | 46.591644 | 0.528302 | 0.002695 | 0.2250 |
| std | 0.801210 | 497.183763 | 0.177034 | 0.149721 | 30.980593 | 45.220324 | 38.855948 | 0.496965 | 53.778815 | 0.499535 | 0.051882 | 0.4179 |
| min | -1.000000 | -1.000000 | 0.000000 | 0.000000 | 15.000000 | 16.000000 | 13.500000 | 0.000000 | -1.000000 | 0.000000 | 0.000000 | 0.0000 |
| 25% | 3.300000 | 1939.000000 | 0.000000 | 0.000000 | 52.000000 | 96.000000 | 73.500000 | 0.000000 | 11.000000 | 0.000000 | 0.000000 | 0.0000 |
| 50% | 3.700000 | 1988.000000 | 0.000000 | 0.000000 | 69.500000 | 124.000000 | 97.500000 | 1.000000 | 24.000000 | 1.000000 | 0.000000 | 0.0000 |
| 75% | 4.000000 | 2007.000000 | 0.000000 | 0.000000 | 91.000000 | 155.000000 | 122.500000 | 1.000000 | 59.000000 | 1.000000 | 0.000000 | 0.0000 |
| max | 5.000000 | 2019.000000 | 1.000000 | 1.000000 | 202.000000 | 306.000000 | 254.000000 | 1.000000 | 276.000000 | 1.000000 | 1.000000 | 1.0000 |

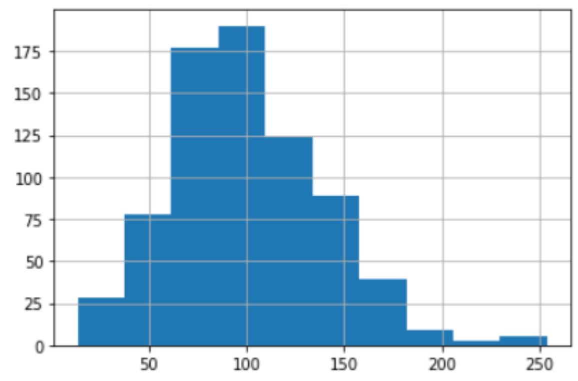
```
df.Rating.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2814125b888>
```



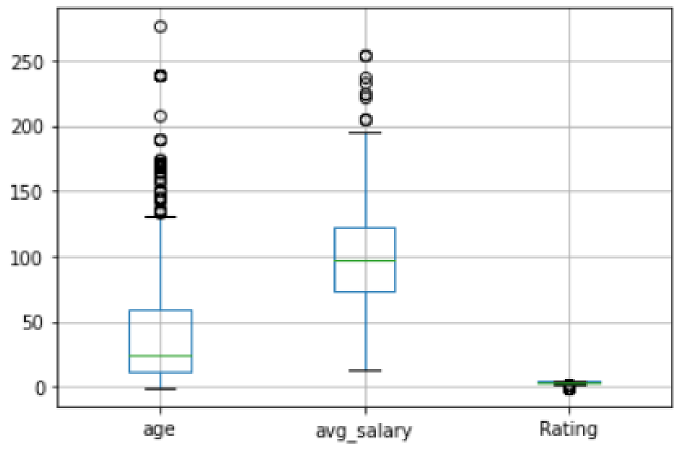
```
df.avg_salary.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28141405a48>
```



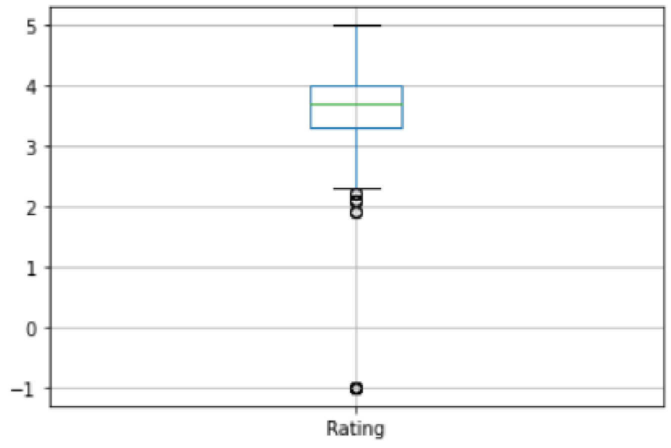
```
df.boxplot(column = ['age','avg_salary','Rating'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x281415baec8>
```



```
df.boxplot(column = 'Rating')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28141703408>
```



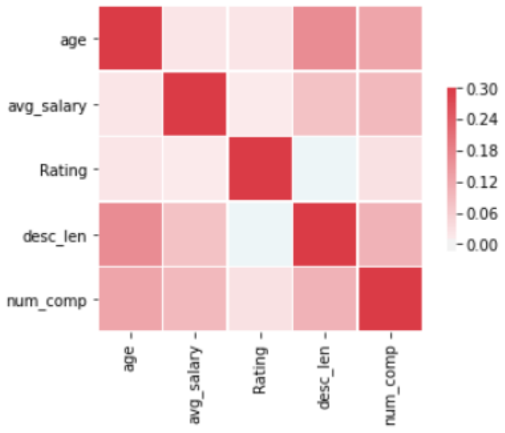
```
: df[['age','avg_salary','Rating','desc_len']].corr()
```

:

| | age | avg_salary | Rating | desc_len |
|---|---|---|---|---|
| age | 1.000000 | 0.019655 | 0.021655 | 0.163911 |
| avg_salary | 0.019655 | 1.000000 | 0.013492 | 0.078808 |
| Rating | 0.021655 | 0.013492 | 1.000000 | -0.012281 |
| desc_len | 0.163911 | 0.078808 | -0.012281 | 1.000000 |

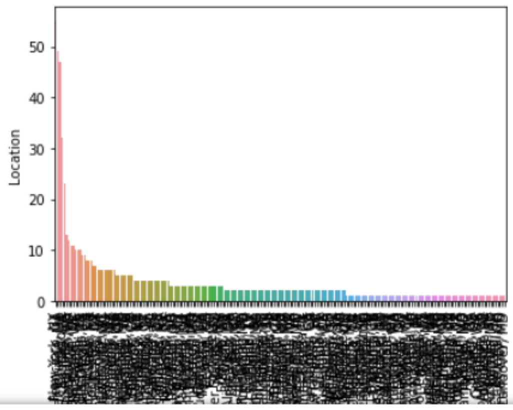
```
: cmap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(df[['age','avg_salary','Rating','desc_len','num_comp']].corr(),vmax=.3, center=0, cmap=cmap,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

```
: <matplotlib.axes._subplots.AxesSubplot at 0x281414022c8>
```



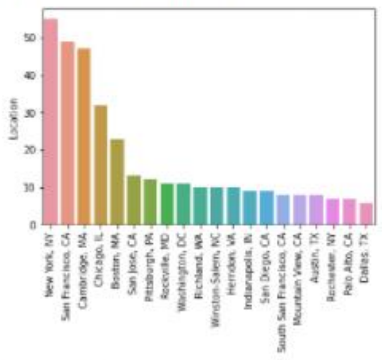
```
In [78]: for i in df_cat.columns:
             cat_num = df_cat[i].value_counts()
             print("graph for %s: total = %d" % (i, len(cat_num)))
             chart = sns.barplot(x=cat_num.index, y=cat_num)
             chart.set_xticklabels(chart.get_xticklabels(), rotation=90)
             plt.show()
```

```
graph for Location: total = 200
```

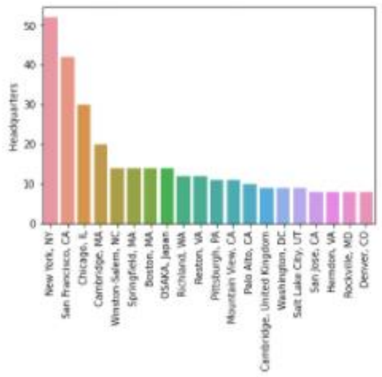


```
for i in df_cat[['Location','Headquarters','company_txt']].columns:
    cat_num = df_cat[i].value_counts()[:20]
    print("graph for %s: total = %d" % (i, len(cat_num)))
    chart = sns.barplot(x=cat_num.index, y=cat_num)
    chart.set_xticklabels(chart.get_xticklabels(), rotation=90)
    plt.show()
```

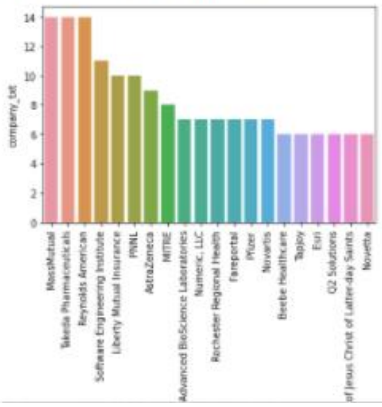
```
graph for Location: total = 20
```



```
graph for Headquarters: total = 20
```



```
graph for company_txt: total = 20
```



```
pd.pivot_table(df, index = 'job_simp', values = 'avg_salary')
```

| job_simp | avg_salary |
|---|---|
| analyst | 65.857843 |
| data engineer | 105.403361 |
| data scientist | 117.564516 |
| director | 168.607143 |
| manager | 84.022727 |
| mle | 126.431818 |
| na | 84.853261 |

```
pd.pivot_table(df, index = ['job_simp','seniority'], values = 'avg_salary')
```

| job_simp | seniority | avg_salary |
|---|---|---|
| analyst | jr | 56.500000 |
| | na | 61.155405 |
| | senior | 79.092593 |
| data engineer | na | 96.701220 |
| | senior | 124.689189 |
| data scientist | jr | 106.500000 |
| | na | 107.043011 |
| | senior | 138.956522 |
| director | na | 168.607143 |
| manager | na | 84.022727 |
| mle | na | 119.133333 |
| | senior | 142.071429 |
| na | na | 73.988189 |
| | senior | 109.061404 |

```
import pandas as pd
```

```
file=r'/Users/Monica/Desktop/salary_prediction_app/eda_data.csv'
```

```
train=pd.read_csv(file)
```

```
train=train.iloc[:,1:]
```

```
import re
```

```
k=[re.sub('[$A-Za-z\(\)\.:]','',x) for x in train['Salary Estimate']]
```

```
train['Salary Estimate']=pd.Series(k).str.split('-',expand=True).astype(float).mean(axis=1)
```

```
train['Company Name']=train['Company Name'].str.split('\n',expand=True).iloc[:,0]
```

```
train.columns
```

```
Index(['Job Title', 'Salary Estimate', 'Job Description', 'Rating',
       'Company Name', 'Location', 'Headquarters', 'Size', 'Founded',
       'Type of ownership', 'Industry', 'Sector', 'Revenue', 'Competitors',
       'hourly', 'employer_provided', 'min_salary', 'max_salary', 'avg_salary',
       'company_txt', 'job_state', 'same_state', 'age', 'python_yn', 'R_yn',
       'spark', 'aws', 'excel', 'job_simp', 'seniority', 'desc_len',
       'num_comp'],
      dtype='object')
```

```
# for simplyfying things , you can later on make use of these if you like
# except the salary vars
drop_cols=['Job Description','seniority','job_simp','min_salary','max_salary','avg_salary','company_txt','desc_len']
```

```
train.drop(drop_cols,1,inplace=True)
```

```
train.shape
```

```
(742, 24)
```

```
: train['Size'].value_counts()
```

```
: 1001 to 5000 employees     150
  501 to 1000 employees      134
  10000+ employees           130
  201 to 500 employees       117
  51 to 200 employees         94
  5001 to 10000 employees     76
  1 to 50 employees           31
  Unknown                      9
  -1                           1
  Name: Size, dtype: int64
```

```
: train.to_csv('simple_data.csv',index=False)
```

```
: from mypipes import *
```

```
: cat_vars=list(train.select_dtypes(['object']).columns)
```

```
: num_vars=list(train.select_dtypes(exclude=['object']).columns)
  num_vars.remove('Salary Estimate')
```

```
: cat_vars
```

```
: ['Job Title',
   'Company Name',
   'Location',
   'Headquarters',
   'Size',
   'Type of ownership',
   'Industry',
   'Sector',
   'Revenue',
   'Competitors',
   'job_state']
```

```
: cat_pipe=Pipeline([
      ('cat_var_select',VarSelector(cat_vars)),
      ('create_dummies',get_dummies_Pipe(74))
  ])
```

```
: data_pipe=FeatureUnion([
      ('num_vars',VarSelector(num_vars)),
      ('cat_data',cat_pipe)
  ])
```

```
: from sklearn.linear_model import LinearRegression
  # fitting a simple model , you can later make something complex and tuned
```

```
: complete_pipe=Pipeline([
      ('data_pipe',data_pipe),
      ('model',LinearRegression(fit_intercept=True))
  ])
```

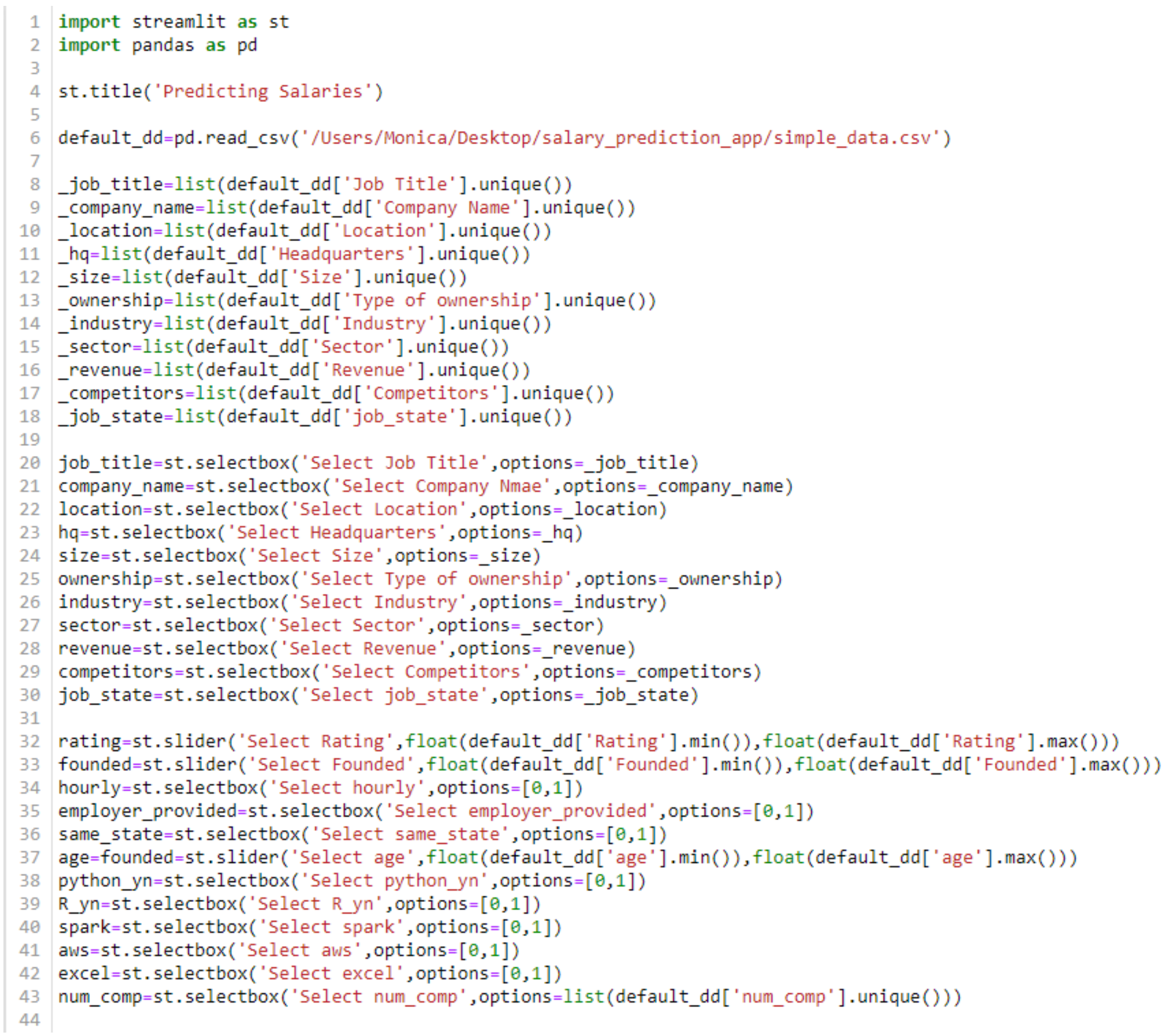
```
: target='Salary Estimate'
  y=train[target]
  x=train.drop(target,1)
```

```
: complete_pipe.fit(x,y)
```

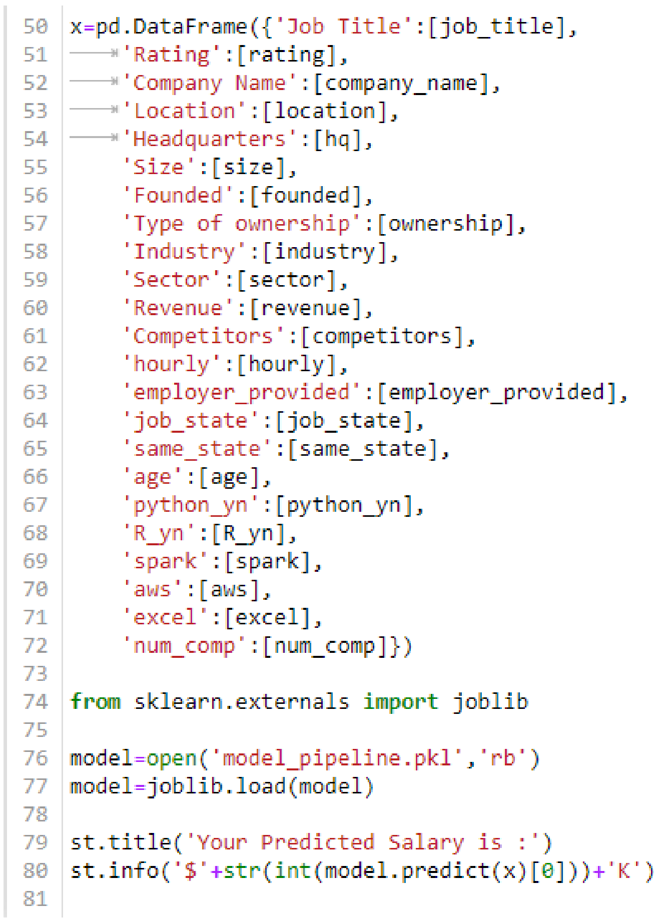
```
: Pipeline(memory=None,
       steps=[('data_pipe', FeatureUnion(n_jobs=None,
         transformer_list=[('num_vars', VarSelector(drop_var=False, var_names=None)), ('cat_data', Pipeline(memory=None,
       steps=[('cat_var_select', VarSelector(drop_var=False, var_names=None)), ('create_dummies', get_dummies_Pipe(freq_cutoff=7
  4))]))],
         transformer_weights=None)), ('model', LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
           normalize=False))])
```

```
: from sklearn.externals import joblib
  joblib.dump(complete_pipe,'model_pipeline.pkl')
```

```
: ['model_pipeline.pkl']
```



```python
import streamlit as st
import pandas as pd

st.title('Predicting Salaries')

default_dd=pd.read_csv('/Users/Monica/Desktop/salary_prediction_app/simple_data.csv')

_job_title=list(default_dd['Job Title'].unique())
_company_name=list(default_dd['Company Name'].unique())
_location=list(default_dd['Location'].unique())
_hq=list(default_dd['Headquarters'].unique())
_size=list(default_dd['Size'].unique())
_ownership=list(default_dd['Type of ownership'].unique())
_industry=list(default_dd['Industry'].unique())
_sector=list(default_dd['Sector'].unique())
_revenue=list(default_dd['Revenue'].unique())
_competitors=list(default_dd['Competitors'].unique())
_job_state=list(default_dd['job_state'].unique())

job_title=st.selectbox('Select Job Title',options=_job_title)
company_name=st.selectbox('Select Company Nmae',options=_company_name)
location=st.selectbox('Select Location',options=_location)
hq=st.selectbox('Select Headquarters',options=_hq)
size=st.selectbox('Select Size',options=_size)
ownership=st.selectbox('Select Type of ownership',options=_ownership)
industry=st.selectbox('Select Industry',options=_industry)
sector=st.selectbox('Select Sector',options=_sector)
revenue=st.selectbox('Select Revenue',options=_revenue)
competitors=st.selectbox('Select Competitors',options=_competitors)
job_state=st.selectbox('Select job_state',options=_job_state)

rating=st.slider('Select Rating',float(default_dd['Rating'].min()),float(default_dd['Rating'].max()))
founded=st.slider('Select Founded',float(default_dd['Founded'].min()),float(default_dd['Founded'].max()))
hourly=st.selectbox('Select hourly',options=[0,1])
employer_provided=st.selectbox('Select employer_provided',options=[0,1])
same_state=st.selectbox('Select same_state',options=[0,1])
age=founded=st.slider('Select age',float(default_dd['age'].min()),float(default_dd['age'].max()))
python_yn=st.selectbox('Select python_yn',options=[0,1])
R_yn=st.selectbox('Select R_yn',options=[0,1])
spark=st.selectbox('Select spark',options=[0,1])
aws=st.selectbox('Select aws',options=[0,1])
excel=st.selectbox('Select excel',options=[0,1])
num_comp=st.selectbox('Select num_comp',options=list(default_dd['num_comp'].unique()))

```



```python
x=pd.DataFrame({'Job Title':[job_title],
    'Rating':[rating],
    'Company Name':[company_name],
    'Location':[location],
    'Headquarters':[hq],
    'Size':[size],
    'Founded':[founded],
    'Type of ownership':[ownership],
    'Industry':[industry],
    'Sector':[sector],
    'Revenue':[revenue],
    'Competitors':[competitors],
    'hourly':[hourly],
    'employer_provided':[employer_provided],
    'job_state':[job_state],
    'same_state':[same_state],
    'age':[age],
    'python_yn':[python_yn],
    'R_yn':[R_yn],
    'spark':[spark],
    'aws':[aws],
    'excel':[excel],
    'num_comp':[num_comp]})

from sklearn.externals import joblib

model=open('model_pipeline.pkl','rb')
model=joblib.load(model)

st.title('Your Predicted Salary is :')
st.info('$'+str(int(model.predict(x)[0]))+'K')

```

## Live Web App Demo Link

Deployment on Heroku:

## Abstract

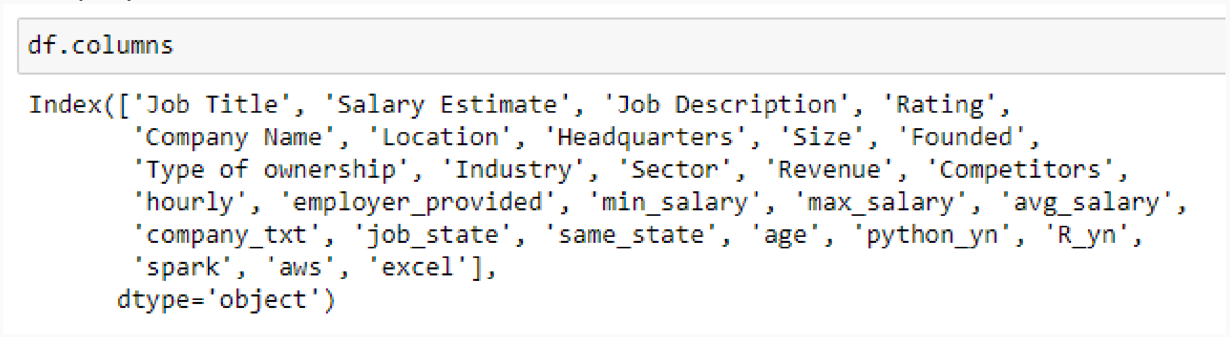
The purpose of this report will be to use the Scraped Data and use this scraped glassdoor salary data to predict salary based on various parameters provided.
This can be used to gain insight into how and why salary varies depending on location and job role etc. This can also be used as a model to gain a marketing advantage, by advertisement targeting those who are more likely to finding jobs in the respective domain to know skillsets required and experience required. Salary Prediction is a regression problem, where using the scraped data and model building will predict what range of salary one can receive.
This is diving into Salary Prediction through Machine Learning Concept.
End to End Project means that it is step by step process, starts with data collection, EDA, Data Preparation which includes cleaning and transforming then selecting, training and saving ML Models, Cross-validation and Hyper-Parameter Tuning and developing web service then Deployment for end users to use it anytime and anywhere.
This repository contains the code for Salary Prediction using python's various libraries.
It used numpy, pandas, matplotlib, seaborn, sklearn, time, joblib, selenium and pickle libraries.
These libraries help to perform individually one particular functionality.
Pandas objects rely heavily on Numpy objects.
Numpy is used for working with arrays. It stands for Numerical Python.
Matplotlib is a plotting library.
Seaborn is data visualization library based on matplotlib.
Sklearn has 100 to 200 models.
"Pickling" is the process whereby a Python object hierarchy is converted into a byte stream.
Time provides many ways of representing time in code, such as objects, numbers, and strings.
Joblib is a set of tools to provide lightweight pipelining in Python.
The purpose of creating this repository is to gain insights into Complete ML Project.
These python libraries raised knowledge in discovering these libraries with practical use of it.
It leads to growth in my ML repository.
These above screenshots and video in Video_File Folder will help you to understand flow of output.

## Motivation

The reason behind building this is, because salary is one of the important factors of job and life and therefore every one of us look at the salary provided by the company before applying or before joining the company. It is also one of the strong factors responsible for performance. When it comes to salary and pay for performance schemes companies need to consider careful what it is they want to reward and in how far there might be different ways to motivate and engage employees. Another reason is that, till now I have worked on individual concepts so I wanted to combine all things that I have learnt till now and create a End to End Project that shows whole life cycle of ML Project. In addition to that, as an employee of a company, I should be able to carry out an entire process own is also essential aspect of it. Building end to end project is gave me wholesome approach to handle given data. Hence, I continue to gain knowledge while practicing the same and spread literary wings in tech-heaven.

## The Data

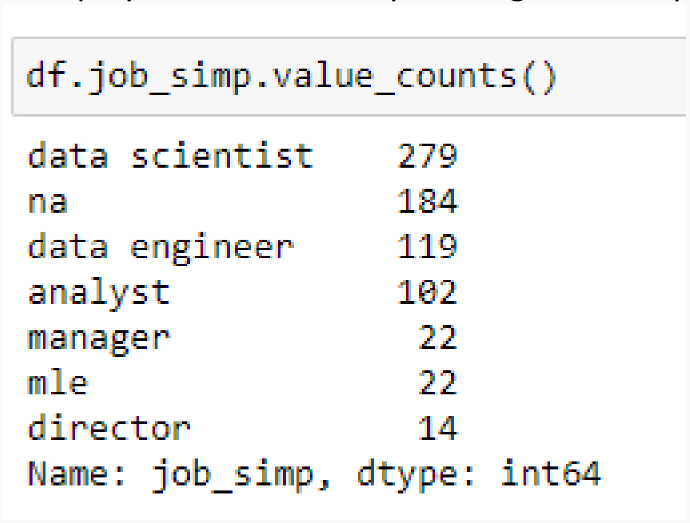
It displays name of the columns.



```
df.columns
```

```
Index(['Job Title', 'Salary Estimate', 'Job Description', 'Rating',
       'Company Name', 'Location', 'Headquarters', 'Size', 'Founded',
       'Type of ownership', 'Industry', 'Sector', 'Revenue', 'Competitors',
       'hourly', 'employer_provided', 'min_salary', 'max_salary', 'avg_salary',
       'company_txt', 'job_state', 'same_state', 'age', 'python_yn', 'R_yn',
       'spark', 'aws', 'excel'],
      dtype='object')
```

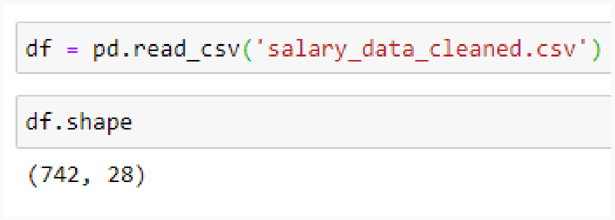
It displays number of unique categories in a particular column.



```
df.job_simp.value_counts()
```

```
data scientist    279
na                184
data engineer     119
analyst           102
manager            22
mle                22
director           14
Name: job_simp, dtype: int64
```

It shows 742 total observations and 28 columns.

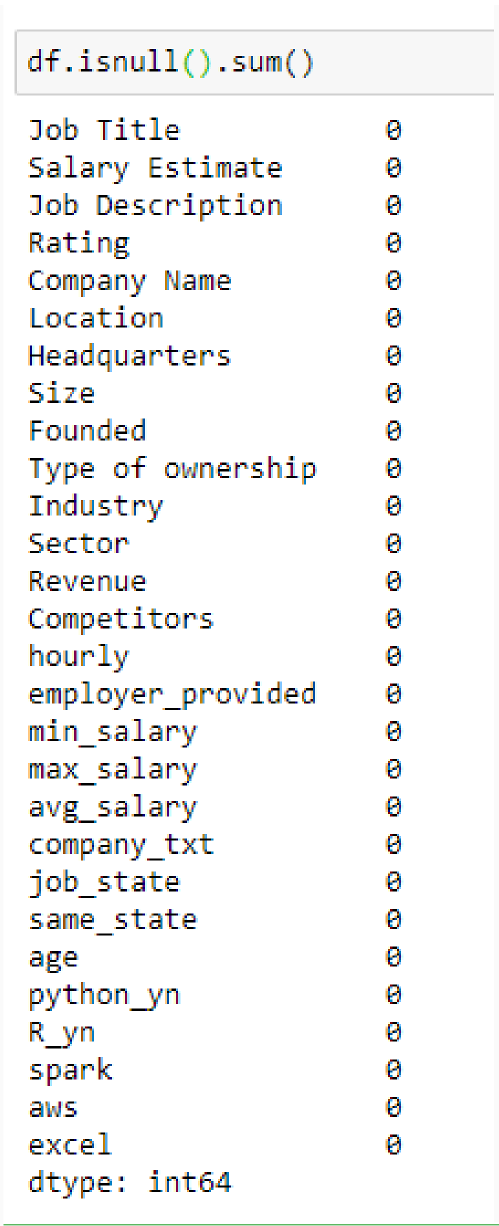


```
df = pd.read_csv('salary_data_cleaned.csv')
```

```
df.shape
```

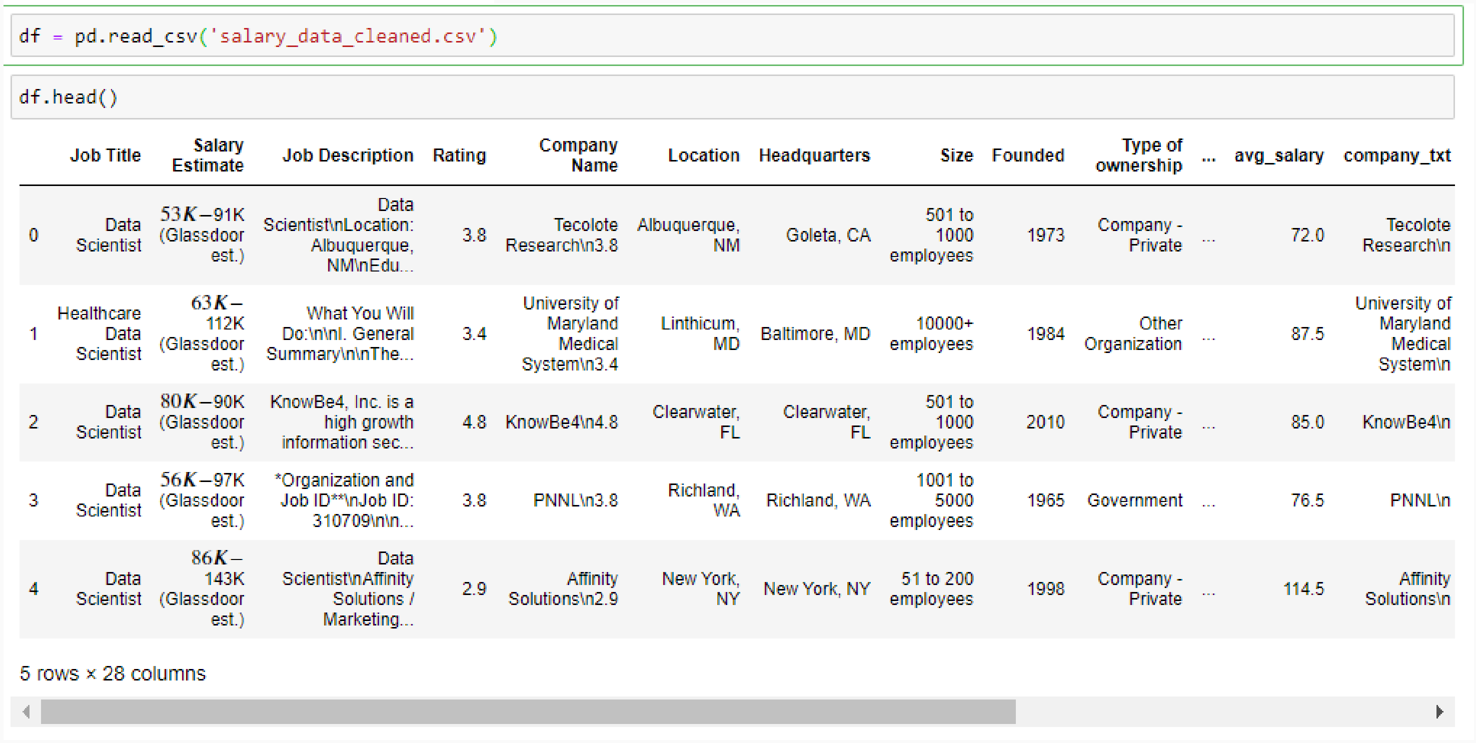
```
(742, 28)
```

It displays missing values if any.



```
df.isnull().sum()
```

```
Job Title            0
Salary Estimate      0
Job Description      0
Rating               0
Company Name         0
Location             0
Headquarters         0
Size                 0
Founded              0
Type of ownership    0
Industry             0
Sector               0
Revenue              0
Competitors          0
hourly               0
employer_provided    0
min_salary           0
max_salary           0
avg_salary           0
company_txt          0
job_state            0
same_state           0
age                  0
python_yn            0
R_yn                 0
spark                0
aws                  0
excel                0
dtype: int64
```



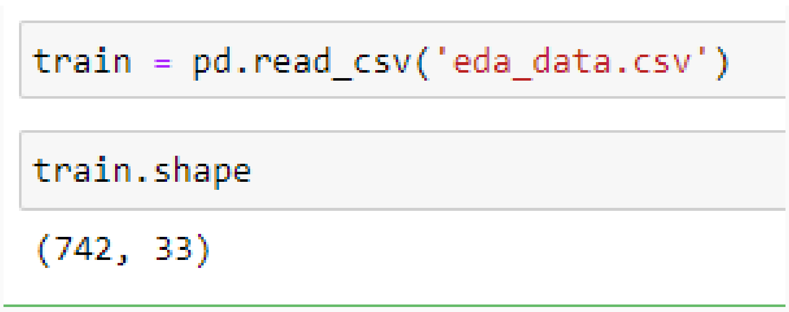
```
df = pd.read_csv('salary_data_cleaned.csv')
```

```
df.head()
```

| | Job Title | Salary Estimate | Job Description | Rating | Company Name | Location | Headquarters | Size | Founded | Type of ownership | ... | avg_salary | company_txt |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Data Scientist | 53*K*−91K (Glassdoor est.) | Data Scientist\nLocation: Albuquerque, NM\nEdu... | 3.8 | Tecolote Research\n3.8 | Albuquerque, NM | Goleta, CA | 501 to 1000 employees | 1973 | Company - Private | ... | 72.0 | Tecolote Research\n |
| 1 | Healthcare Data Scientist | 63*K*−112K (Glassdoor est.) | What You Will Do:\n\nI. General Summary\n\nThe... | 3.4 | University of Maryland Medical System\n3.4 | Linthicum, MD | Baltimore, MD | 10000+ employees | 1984 | Other Organization | ... | 87.5 | University of Maryland Medical System\n |
| 2 | Data Scientist | 80*K*−90K (Glassdoor est.) | KnowBe4, Inc. is a high growth information sec... | 4.8 | KnowBe4\n4.8 | Clearwater, FL | Clearwater, FL | 501 to 1000 employees | 2010 | Company - Private | ... | 85.0 | KnowBe4\n |
| 3 | Data Scientist | 56*K*−97K (Glassdoor est.) | *Organization and Job ID**\nJob ID: 310709\n\n... | 3.8 | PNNL\n3.8 | Richland, WA | Richland, WA | 1001 to 5000 employees | 1965 | Government | ... | 76.5 | PNNL\n |
| 4 | Data Scientist | 86*K*−143K (Glassdoor est.) | Data Scientist\nAffinity Solutions / Marketing... | 2.9 | Affinity Solutions\n2.9 | New York, NY | New York, NY | 51 to 200 employees | 1998 | Company - Private | ... | 114.5 | Affinity Solutions\n |

5 rows × 28 columns

It displays number of rows and columns.

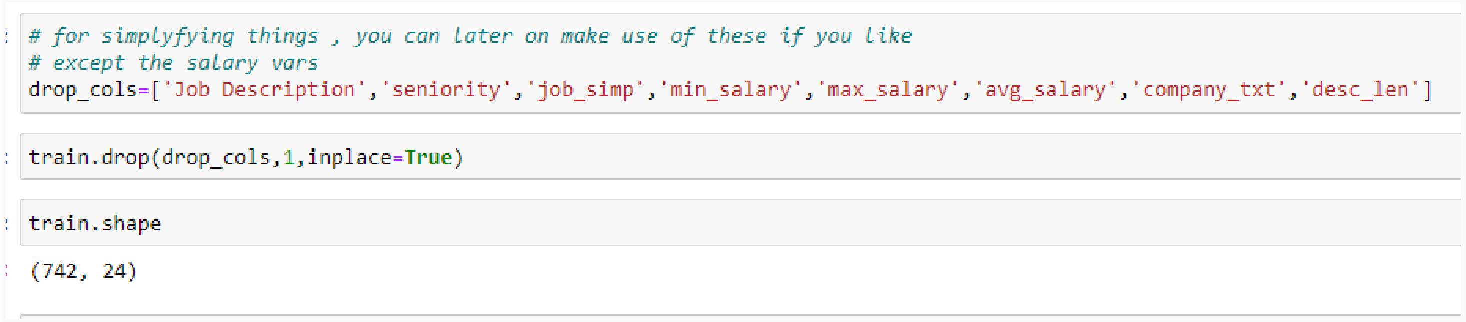


```
train = pd.read_csv('eda_data.csv')
```

```
train.shape
```

```
(742, 33)
```

Dropped not required columns and then displays final number of rows and columns used.



```
# for simplyfying things , you can later on make use of these if you like
# except the salary vars
drop_cols=['Job Description','seniority','job_simp','min_salary','max_salary','avg_salary','company_txt','desc_len']
```

```
train.drop(drop_cols,1,inplace=True)
```

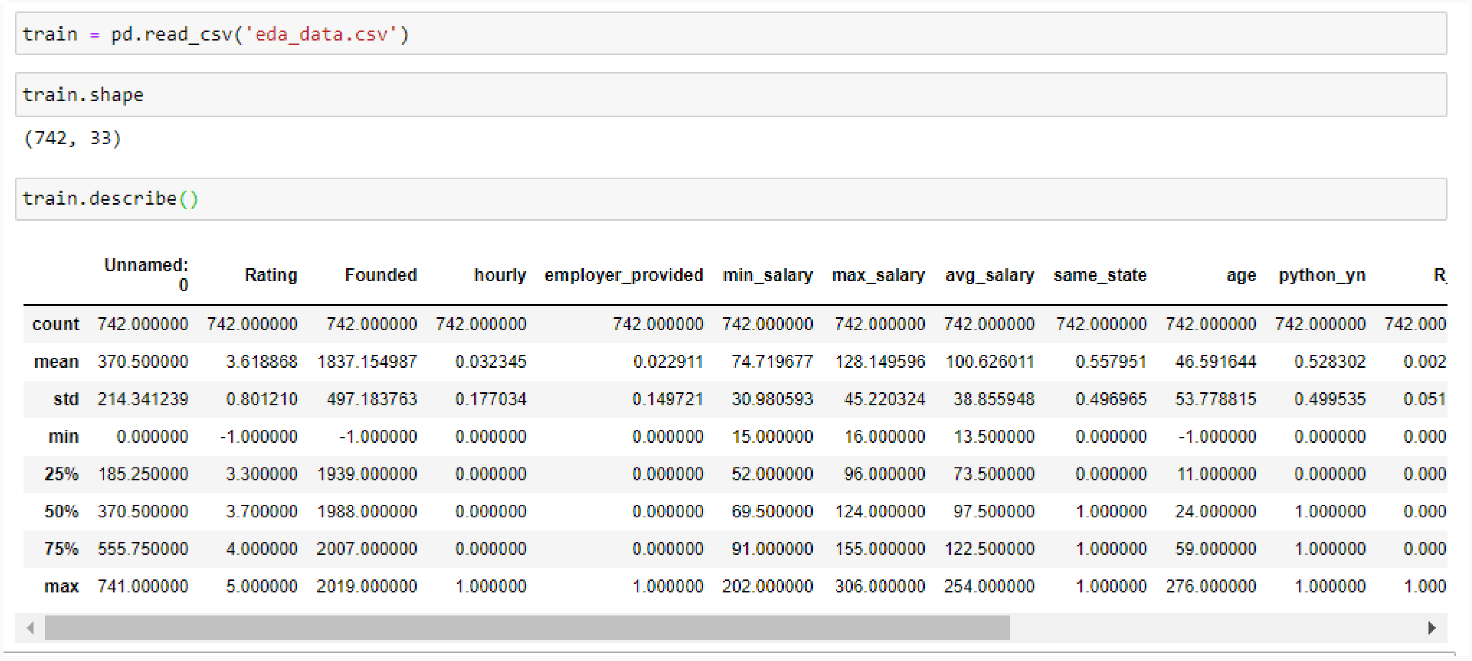
```
train.shape
```

```
(742, 24)
```

## Analysis of the Data

Let's start by doing a general analysis of the data as a whole, including all the features the Linear Regression algorithm will be using.

- Basic Statistics



```
train = pd.read_csv('eda_data.csv')
```

```
train.shape
```

```
(742, 33)
```

```
train.describe()
```

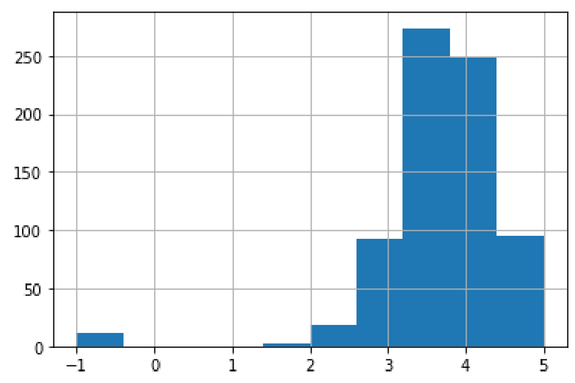
| | Unnamed: 0 | Rating | Founded | hourly | employer_provided | min_salary | max_salary | avg_salary | same_state | age | python_yn | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000000 | 742.000 |
| **mean** | 370.500000 | 3.618868 | 1837.154987 | 0.032345 | 0.022911 | 74.719677 | 128.149596 | 100.626011 | 0.557951 | 46.591644 | 0.528302 | 0.002 |
| **std** | 214.341239 | 0.801210 | 497.183763 | 0.177034 | 0.149721 | 30.980593 | 45.220324 | 38.855948 | 0.496965 | 53.778815 | 0.499535 | 0.051 |
| **min** | 0.000000 | -1.000000 | -1.000000 | 0.000000 | 0.000000 | 15.000000 | 16.000000 | 13.500000 | 0.000000 | -1.000000 | 0.000000 | 0.000 |
| **25%** | 185.250000 | 3.300000 | 1939.000000 | 0.000000 | 0.000000 | 52.000000 | 96.000000 | 73.500000 | 0.000000 | 11.000000 | 0.000000 | 0.000 |
| **50%** | 370.500000 | 3.700000 | 1988.000000 | 0.000000 | 0.000000 | 69.500000 | 124.000000 | 97.500000 | 1.000000 | 24.000000 | 1.000000 | 0.000 |
| **75%** | 555.750000 | 4.000000 | 2007.000000 | 0.000000 | 0.000000 | 91.000000 | 155.000000 | 122.500000 | 1.000000 | 59.000000 | 1.000000 | 0.000 |
| **max** | 741.000000 | 5.000000 | 2019.000000 | 1.000000 | 1.000000 | 202.000000 | 306.000000 | 254.000000 | 1.000000 | 276.000000 | 1.000000 | 1.000 |

- Graphing of Features

Graph Set 1

```
df.Rating.hist()
```

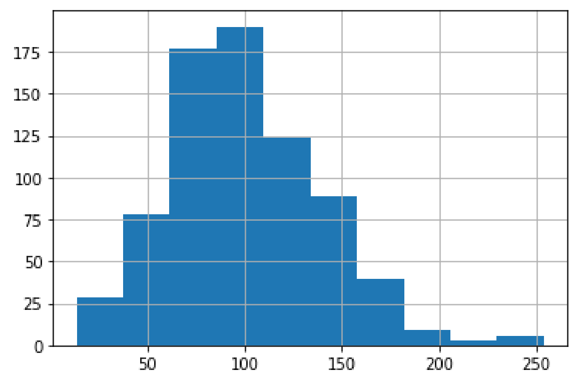
```
<matplotlib.axes._subplots.AxesSubplot at 0x2814125b888>
```



Graph Set 2

```
df.avg_salary.hist()
```

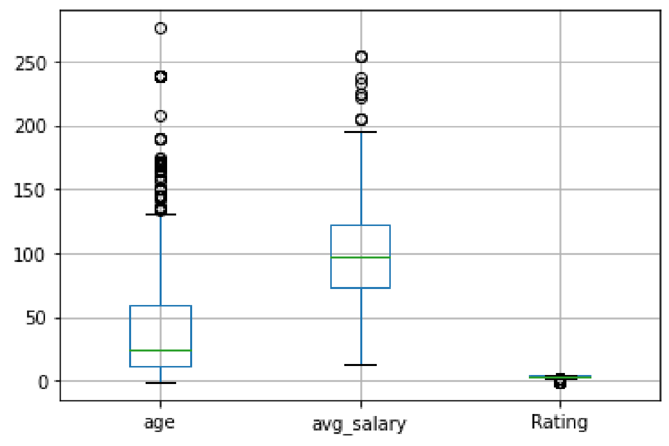
```
<matplotlib.axes._subplots.AxesSubplot at 0x28141405a48>
```



Graph Set 3

```
df.boxplot(column = ['age','avg_salary','Rating'])
```

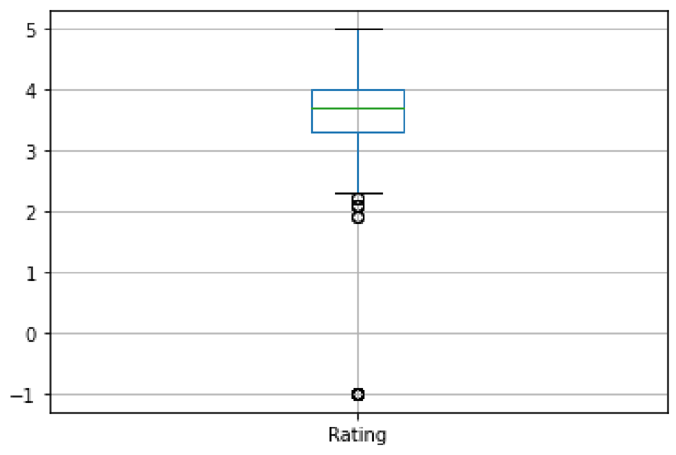
```
<matplotlib.axes._subplots.AxesSubplot at 0x281415baec8>
```



## Graph Set 4

```
df.boxplot(column = 'Rating')
```

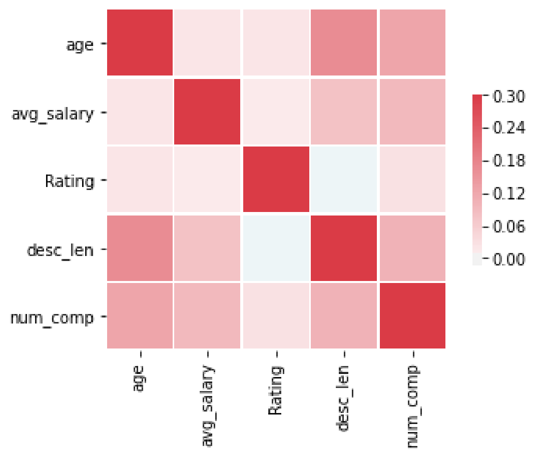
```
<matplotlib.axes._subplots.AxesSubplot at 0x28141703408>
```



## Graph Set 5

```
cmap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(df[['age','avg_salary','Rating','desc_len','num_comp']].corr(),vmax=.3, center=0, cmap=cmap,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

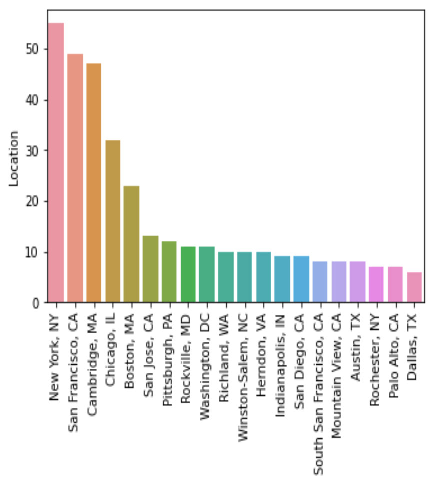
```
<matplotlib.axes._subplots.AxesSubplot at 0x281414022c8>
```



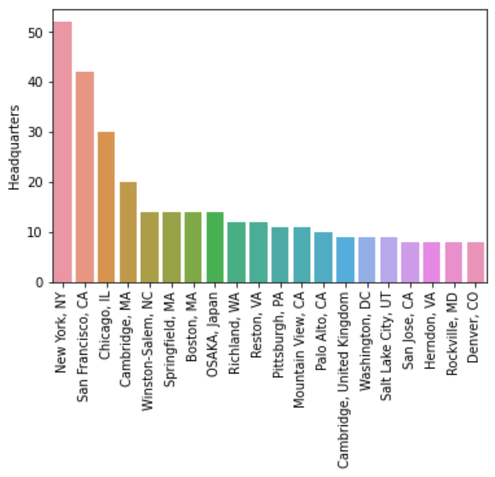
## Graph Set 6

```
for i in df_cat[['Location','Headquarters','company_txt']].columns:
    cat_num = df_cat[i].value_counts()[:20]
    print("graph for %s: total = %d" % (i, len(cat_num)))
    chart = sns.barplot(x=cat_num.index, y=cat_num)
    chart.set_xticklabels(chart.get_xticklabels(), rotation=90)
    plt.show()
```

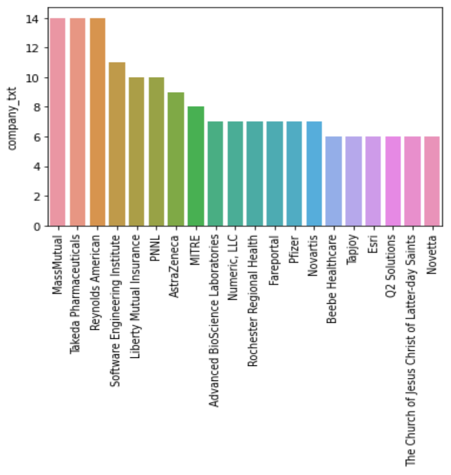
```
graph for Location: total = 20
```



```
graph for Headquarters: total = 20
```



```
graph for company_txt: total = 20
```



Modelling

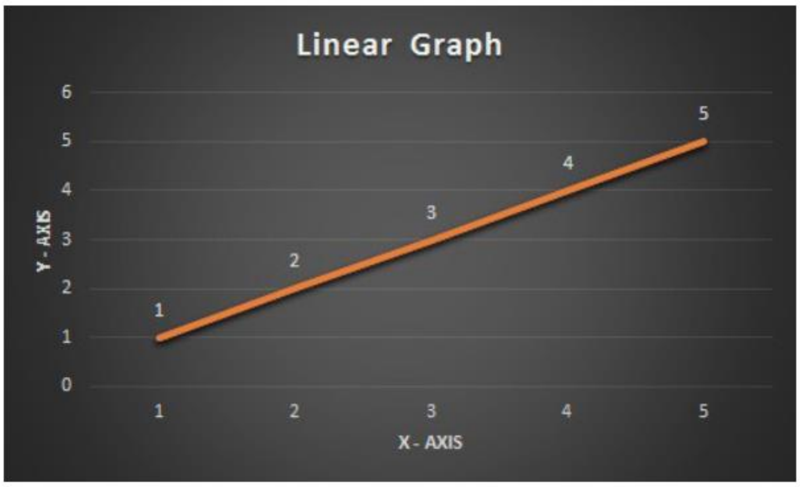
- Math behind the metrics

Linear Regression is a predictive algorithm which provides a Linear relationship between *Prediction* (Call it *'Y'*) and Input (Call is *'X'*).

As we know from the basic maths that if we plot an 'X','Y' graph, a linear relationship will always come up with a straight line. For example, if we plot the graph of these values.

```
(Input) X = 1,2,3,4,5
(Prediction) Y = 1,2,3,4,5
```

It will be a perfectly straight line



Linear Straight Line graph

## Equation of Straight Line from 2 Points

The equation of a straight line is written using the `y = mx + b`, where `m` is the slope (Gradient) and `b` is y-intercept (where the line crosses the Y axis).

Once we get the equation of a straight line from 2 points in space in `y = mx + b` format, we can use the same equation to predict the points at different values of `x` which result in a straight line.

In this formula, `m` is the slope and `b` is y-intercept.

> *Linear regression is a way to predict the* `'Y'` *values for unknown values of Input* `'X'` *like* `1.5, 0.4, 3.6, 5.7` *and even for* `-1, -5, 10` *etc.*

## Linear Regression Formula Analyses

The equation is given by:

y=a+bx

a and b can be computed by the following formulas:

b =
$$\frac{n\sum xy-(\sum x)(\sum y)}{n\sum x^2-(\sum x)^2}$$

a = $\frac{\sum y-b(\sum x)}{n}$

Where,

x and y are the variables for which we will make regression line.

- b = Slope of the line.
- a = Y-intercept of the line.
- X = Values of the first data set.
- Y = Values of the second data set.

Q.1: Find out the linear regression equation for the following sets of data:

| | | | | |
|---|---|---|---|---|
| X | 2 | 3 | 5 | 8 |
| Y | 3 | 6 | 5 | 12 |

Solution:

| X | Y | X^2 | XY |
|---|---|---|---|
| 2 | 3 | 4 | 6 |
| 3 | 6 | 9 | 18 |
| 5 | 5 | 25 | 25 |
| 8 | 12 | 64 | 96 |
| $\sum X$ =18 | $\sum Y = 26$ | $\sum X^2 = 102$ | $\sum XY = 145$ |

Now we will apply the formula to get values of a and b, as follows:

$b = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$

$b = \frac{4\times145 - 18\times26}{4\times102 - 324}$

$b = \frac{112}{84}$

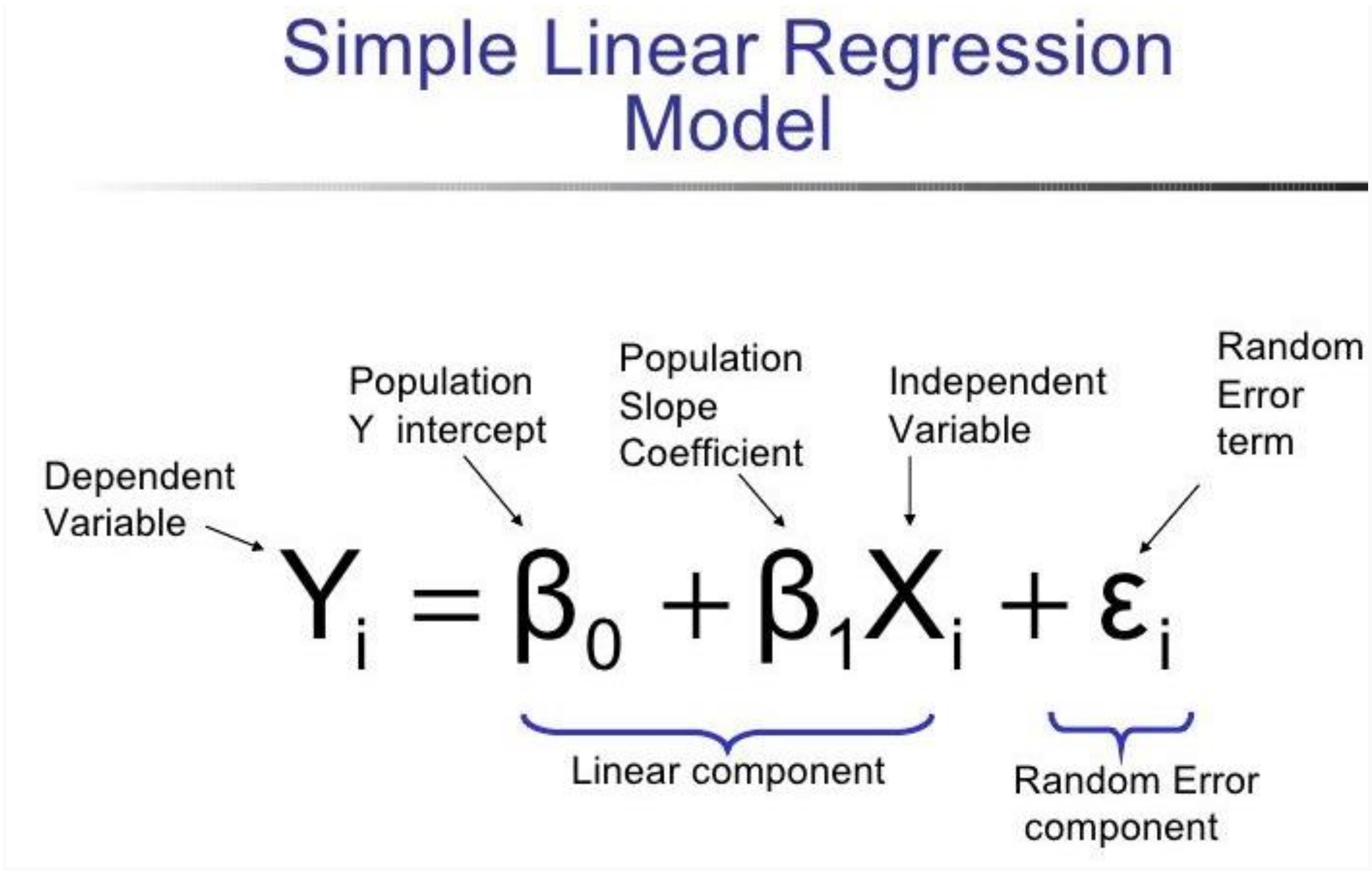
b = 1.33

$a = \frac{\sum y - b(\sum x)}{n}$
$= \frac{26 - 1.33\times18}{4}$

a = 0.515

Linear regression equation is given by:

y = 0.515 + 1.33x

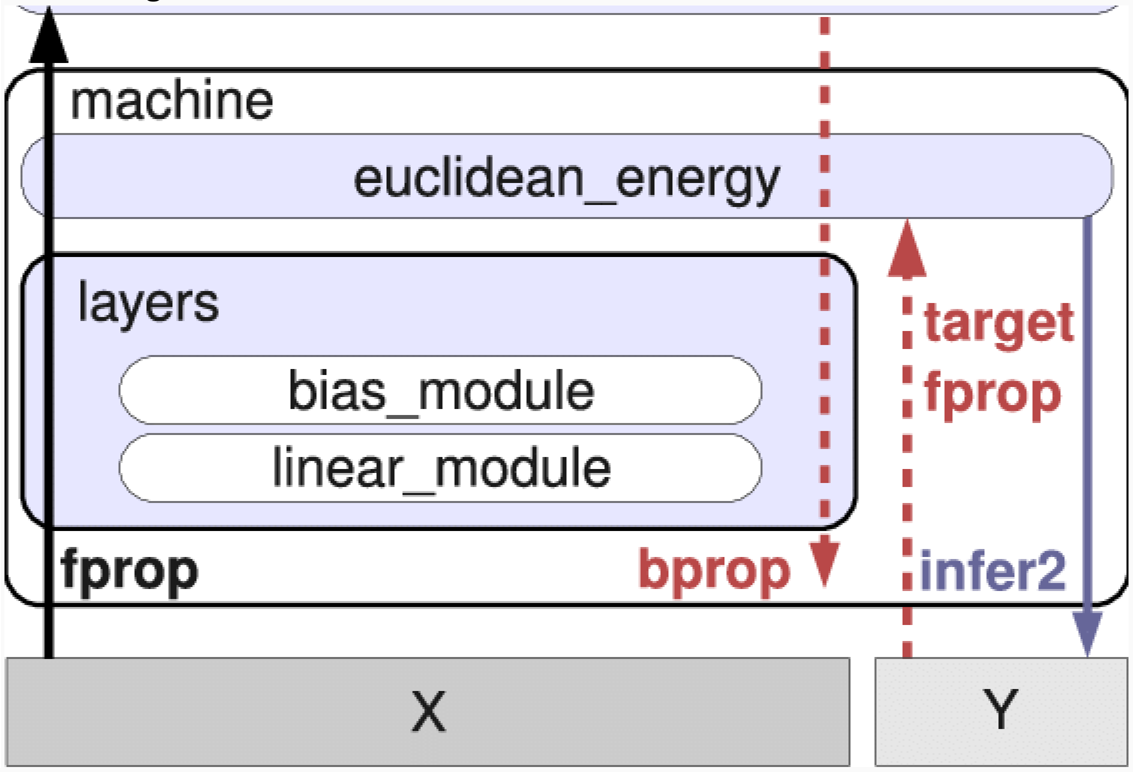


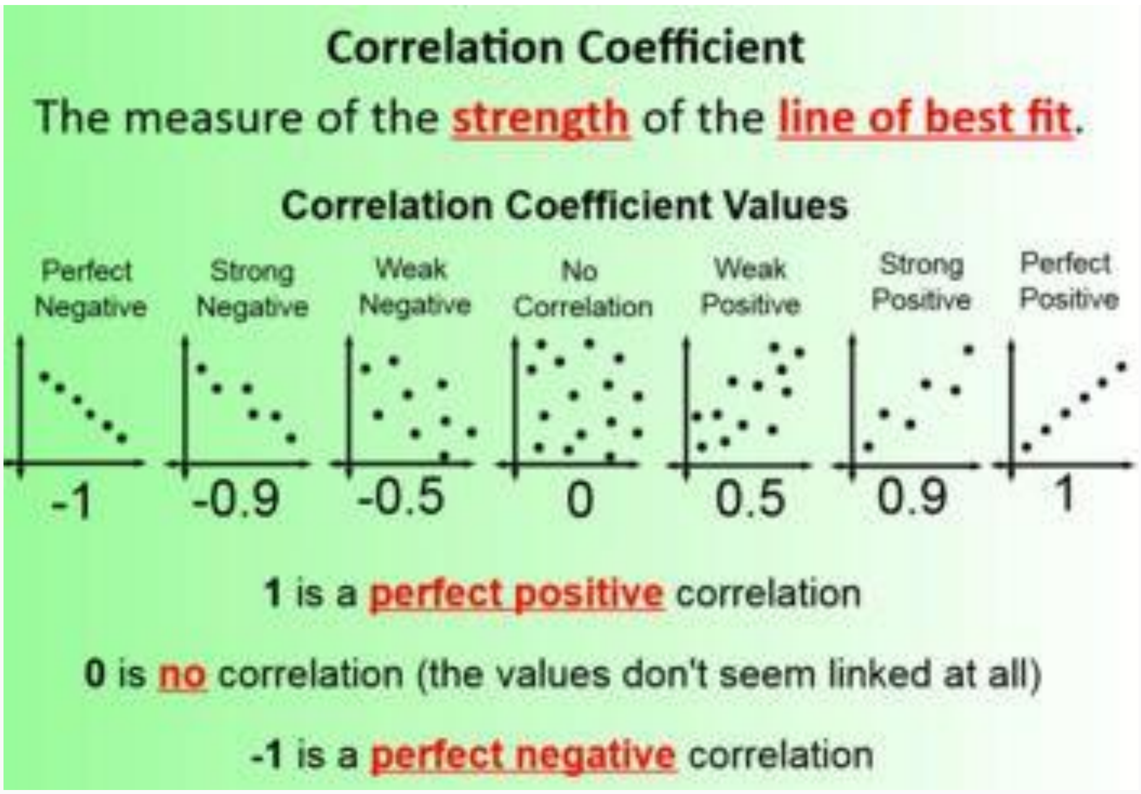
## Simple Linear Regression Model

$$Y_i = \underbrace{\beta_0 + \beta_1 X_i}_{\text{Linear component}} + \underbrace{\varepsilon_i}_{\text{Random Error component}}$$

- Model Architecture Process Through Visualization

Linear Regression Architecture:





Correlation Coefficient

The measure of the strength of the line of best fit.

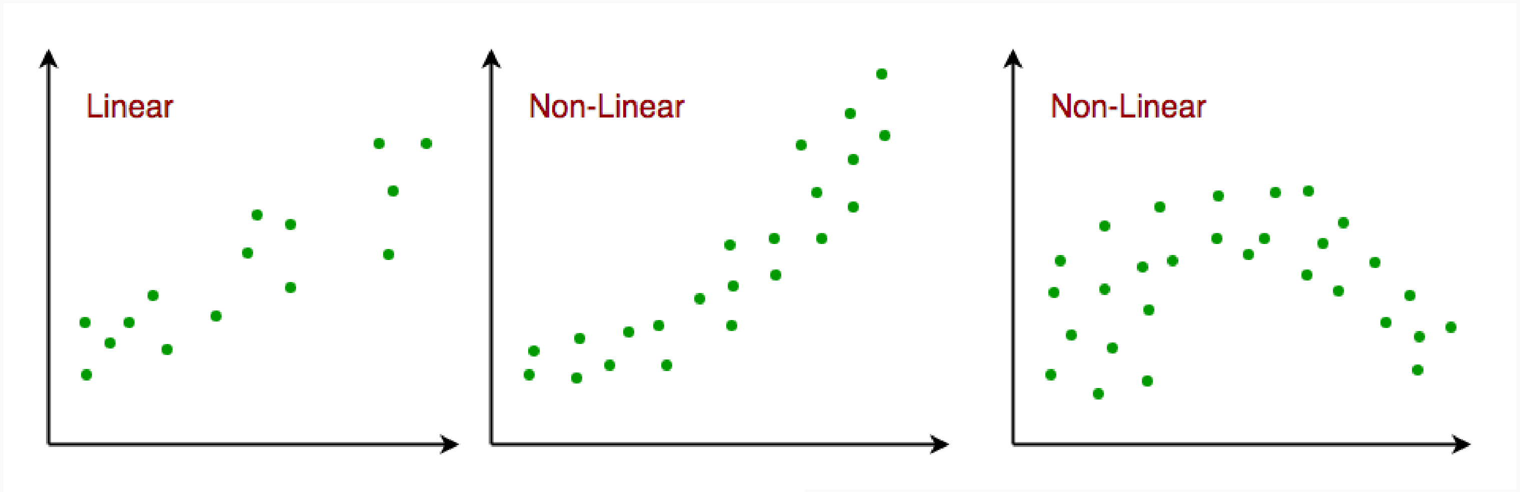
Correlation Coefficient Values

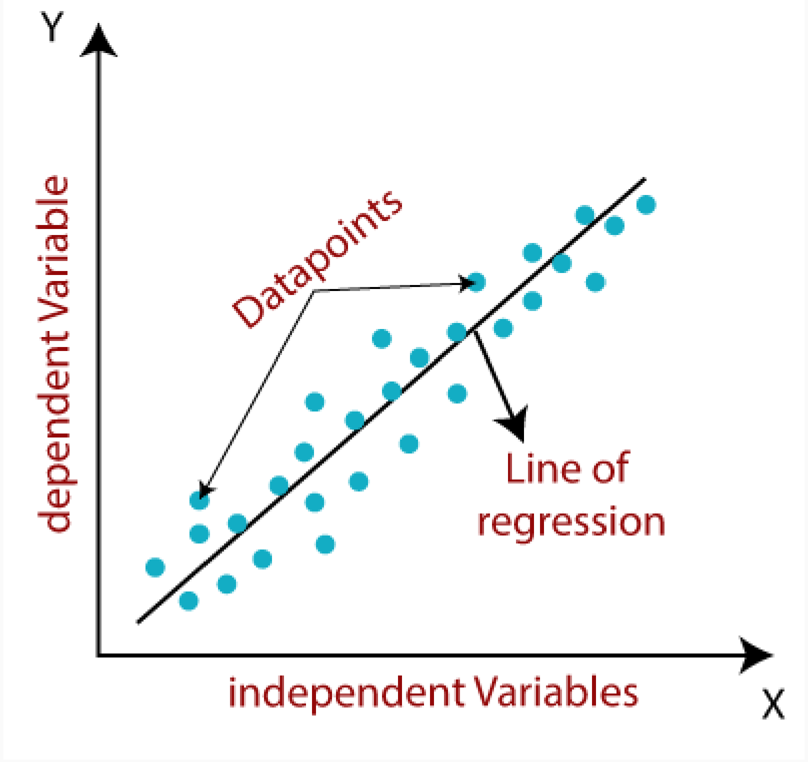
Perfect Negative
Strong Negative
Weak Negative
No Correlation
Weak Positive
Strong Positive
Perfect Positive
-1
-0.9
-0.5
0
0.5
0.9
1

1 is a perfect positive correlation

0 is no correlation (the values don't seem linked at all)

-1 is a perfect negative correlation





- Quick Notes

Step 1: Scraped the data of glassdoor (glassdoor_scraper.py)and collected data in glassdoor_jobs.csv file.

- Initialized web driver.
- Loaded the page.
- Tested the "sign in" prompt and got rid of it.
- See through list of job.
- Set to "not found" value.
- Converted dictionary objects into pandas dataframe.
- Collected data in csv file.

Step 2: Performed data cleaning (data_cleaning.py)on glassdoor_jobs.csv and created salary_data_cleaned.csv file.

- Read stored csv file to clean the data.
- Performed salary parsing.
- Performed operation like 'apply' on columns such as "company name", "state", "age" etc and stored data in csv file.

Step 3: Performed EDA (data_eda.ipynb) on salary_data_cleaned.csv and created eda_data.csv file.

- Read the cleaned data csv file.
- Analyzed the data through ".columns, .value_counts" and other operations.
- Visualized data through histograms, box plots and heat maps.

Step 4: Built model (model_building.ipynb) on eda_data.csv and created simple_data.csv file and model_pipeline.pkl.

- Read the eda csv file.
- Dropped columns those were not required.
- Created and stored data in new csv file.
- Created dummy variables using ".get_dummies()" function.
- Created pipeline and Fitted a linear model on trained data and for the same.
- Saved the model as pickle file to re-use it.

Step 5: Generated web service in streamlit (salary_prediction.py) on simple_data.csv  file and model_pipeline.pkl file.

- Created data frame for required fields. And then predicted the salary value.

- The Model Analysis

<u>Scraped and collected data in csv file</u> – First, initialized web driver. WebDriver is an interface. An interface contains empty methods that have been defined but not implemented. And, since it has empty methods you won't actually need to instantiate it and so you cannot instantiate it. WebDriver is an interface provided by Selenium WebDriver. As we know that interfaces in Java are the collection of constants and abstract methods (methods without any implementation).
The WebDriver interface serves as a contract that each browser-specific implementation like ChromeDriver, FireFoxDriver must follow. The interface allows sending a message to an object without concerning which classes it belongs. Class needs to provide functionality for the methods declared in the interface. Interfaces are useful because they provide contracts that objects can use to work together without needing to know anything else about each other. The point of interfaces is not to help you remember what method to implement, it is here to define a contract. Second, loaded the page. Then, tested the "sign in" prompt and got rid of it. Fourth, see through list of job. Fifth, set to "not found" value and it is an important step. Sixth, converted dictionary objects into pandas data frame so that can store collected data in csv file.

Cleaned the collected data – Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mis-labeled. To perform a Python data cleansing, you can drop the missing values, replace them, replace each NaN with a scalar value, or fill forward or backward. First, read stored csv file to clean the data. The parser itself is created from a grammar specification defined in the file Grammar/Grammar in the standard Python distribution. The parse trees stored in the ST objects created by this module are the actual output from the internal parser. A parser is a software component that takes input data (frequently text) and builds a data structure – often some kind of parse tree, abstract syntax tree or other hierarchical structure, giving a structural representation of the input while checking for correct syntax. Second, performed salary parsing. Third, performed operation like 'apply' on columns such as "company name", "state", "age" etc and stored data in csv file.
Performed EDA – The primary goal of EDA is to maximize the analyst's insight into a data set and into the underlying structure of a data set, while providing all of the specific items that an analyst would want to extract from a data set, such as: a good-fitting, parsimonious model and a list of outliers. There are many libraries available in python like pandas, NumPy, matplotlib, seaborn to perform EDA. The four types of EDA are univariate non-graphical, multivariate non- graphical, univariate graphical, and multivariate graphical. First, read the cleaned data csv file. Second, analysed the data through ".columns, .value_counts" and other operations. Third, plotted data through histograms, box plots and heat maps. A correlation heatmap uses colored cells, typically in a monochromatic scale, to show a 2D correlation matrix (table) between two discrete dimensions. Correlation ranges from -1 to +1. Values closer to zero means there is no linear trend between the two variables. The close to 1 the correlation is the more positively correlated they are; that is as one increases so does the other and the closer to 1 the stronger this relationship is. A box plot is a method for graphically depicting groups of numerical data through their quartiles. The box extends from the Q1 to Q3 quartile values of the data, with a line at the median (Q2). The whiskers extend from the edges of box to show the range of the data. A box and whisker plot is a way of summarizing a set of data measured on an interval scale. It is often used in explanatory data analysis. This type of graph is used to show the shape of the distribution, its central value, and its variability. a histogram is representation of the distribution of numerical data, where the data are binned and the count for each bin is represented. More generally, in plotly a histogram is an aggregated bar chart, with several possible aggregation functions (e.g. sum, average, count). The purpose of a histogram (Chambers) is to graphically summarize the distribution of a univariate data set. It is used to summarize discrete or continuous data that are measured on an interval scale.
Built model – First, read the eda csv file. Second, dropped columns those were not required. Third, created and stored data in new csv file. '.get_dummies' will convert your categorical string values into dummy variables. 'pd.get_dummies' create a new data frame containing unique values as columns which consists of zeros and ones. Fourth, created dummy variables using ".get_dummies()" function. It provides end-to-end velocity by eliminating errors and combatting bottlenecks or latency. It can process multiple data streams at once. In short, it is an absolute necessity for today's data-driven enterprise. A data pipeline views all data as streaming data and it allows for flexible schemas. A pipeline consists of a sequence of stages. There are two basic types of pipeline stages: Transformer and Estimator. A Transformer takes a dataset as input and produces an augmented dataset as output. Fifth, created pipeline and fitted a linear model on trained data and for the same. The fit() method takes the training data as arguments, which can be one array in

the case of unsupervised learning, or two arrays in the case of supervised learning. Sixth, saved the model as pickle file to re-use it.
Generated web app – Built web app in streamlit for end-users.
Challenge that I faced in this project was while scrapping data and to automate the things with selenium and secondly while building web app when I used request.get() could work as it should.

## Creation of App

Here, I am created Streamlit App. Created data frame for required fields. And then predicted the salary value.

## Technical Aspect

Numpy used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. It contains a multi-dimensional array and matrix data structures. It can be utilised to perform a number of mathematical operations on arrays.

Pandas module mainly works with the tabular data. It contains Data Frame and Series. Pandas is 18 to 20 times slower than Numpy. Pandas is seriously a game changer when it comes to cleaning, transforming, manipulating and analyzing data.

Matplotlib is used for EDA. Visualization of graphs helps to understand data in better way than numbers in table format. Matplotlib is mainly deployed for basic plotting. It consists of bars, pies, lines, scatter plots and so on. Inline command display visualization inline within frontends like in Jupyter Notebook, directly below the code cell that produced it.

Seaborn provides a high-level interface for drawing attractive and informative statistical graphics. It provides a variety of visualization patterns and visualize random distributions.

Sklearn is known as scikit learn. It provides many ML libraries and algorithms for it. It provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

Need to train_test_split - Using the same dataset for both training and testing leaves room for miscalculations, thus increases the chances of inaccurate predictions.
The train_test_split function allows you to break a dataset with ease while pursuing an ideal model. Also, keep in mind that your model should not be overfitting or underfitting.

Joblib is used in order to identify the location of the program to be executed in a JCL.
The JOBLIB statement is specified after the JOB statement and before the EXEC statement. This can be used only for the stream procedures and programs. Joblib is a set of tools to provide lightweight pipelining in Python. In particular: transparent disk-caching of functions and lazy re-evaluation (memoize pattern) easy simple parallel computing. Joblib is optimized to be fast and robust in particular on large data and has specific optimizations for numpy arrays.

Selenium is a strong set of tools that firmly supports the quick development of test automation of web applications. It offers a set of testing functions that are specially designed to the requirements of testing of a web application. The structured automation testing life cycle comprises of a multistage process that supports the activities required to utilize and introduce an automated test tool, develop and run test cases, develop test design, build and handle test data and environment.

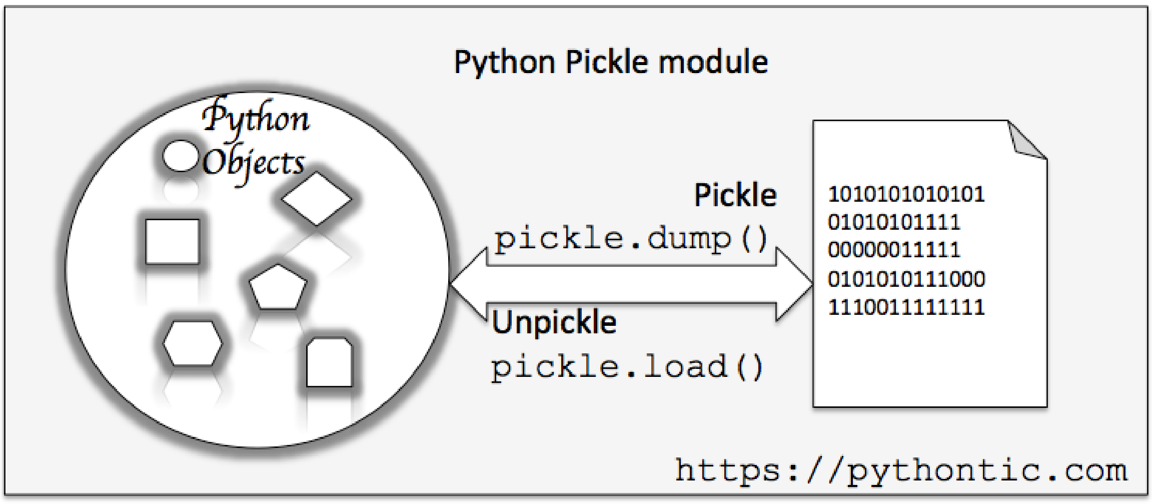
Time provides functionality other than representing time, like waiting during code execution and measuring the efficiency of your code.

Wordcloud is a visual representation of text data. It displays a list of words, the importance of each beeing shown with font size or color. This format is useful for quickly perceiving the most prominent terms.

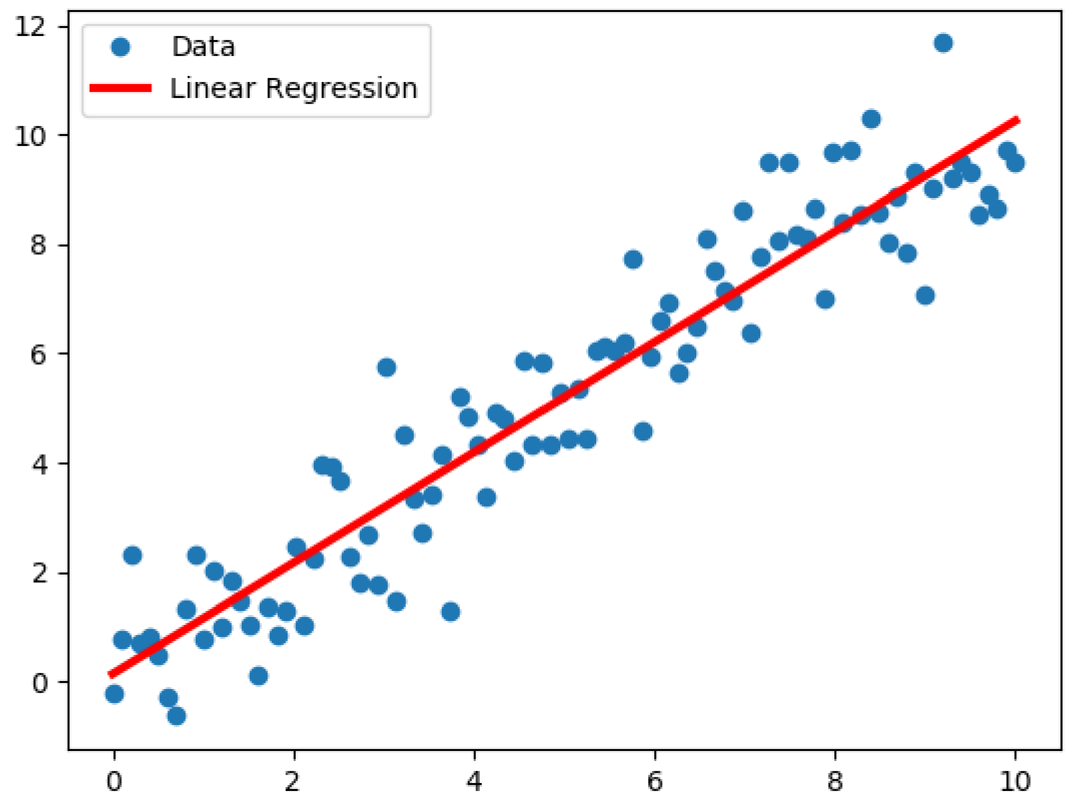
Nltk library in Natural Language Processing with Python provides a practical introduction to programming for language processing. The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries.

Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. In just a few minutes you can build and deploy powerful data apps. It is a very easy library to create a perfect dashboard by spending a little amount of time. It also comes with the inbuilt webserver and lets you deploy in the docker container. When you run the app, the localhost server will open in your browser automatically.

Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.



Linear regression is the next step up after correlation. It is used when we want to predict the value of a variable based on the value of another variable. The variable we want to predict is called the dependent variable. Because the model is based on the equation of a straight line, y=a+bx, where a is the y-intercept (the value of y when x=0) and b is the slope (the degree to which y increases as x increases one unit). Linear regression plots a straight line through a y vs. x scatterplot. That is why it is call linear regression. Simple linear regression is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables: One variable, denoted x, is regarded as the predictor, explanatory, or independent variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable. In essence, multiple regression is the extension of ordinary least-squares (OLS) regression that involves more than one explanatory variable.



## Linear versus Logistic Regression

| ■Linear Regression | ■Logistic Regression |
|---|---|
| ■Target is an interval variable.<br><br>■Input variables have any measurement level.<br><br>■Predicted values are the mean of the target variable at the given values of the input variables. | ■Target is a discrete (binary or ordinal) variable.<br><br>■Input variables have any measurement level.<br><br>■Predicted values are the probability of a particular level(s) of the target variable at the given values of the input variables. |

Installation

Using intel core i5 9th generation with NVIDIA GFORECE GTX1650.
Windows 10 Environment Used.
Already Installed Anaconda Navigator for Python 3.x
The Code is written in Python 3.8.
If you don't have Python installed you can install Python from its official site.
If you are using a lower version of Python you can upgrade using the pip package, ensuring you have the latest version of pip, *python* -m pip install --*upgrade pip and press Enter.*

## Run/How to Use/Steps

Keep your internet connection on while running or accessing files and throughout too.
Follow this when you want to perform from scratch.
Open Anaconda Prompt, Perform the following steps:
cd <PATH>
pip install pandas
pip install matplotlib
pip install seaborn
pip install numpy
Note: If it shows error as 'No Module Found' , then install relevant module.
You can also create requirement.txt file as, pip freeze > requirements.txt
cd <PATH-TO-FOLDER>
run .py or .ipynb files.
Paste URL to browser to check whether working locally or not.

Follow this when you want to just perform on local machine.
Download ZIP File.
Right-Click on ZIP file in download section and select Extract file option, which will unzip file.
Move unzip folder to desired folder/location be it D drive or desktop etc.
Open Anaconda Prompt, write cd <PATH> and press Enter.
eg: cd C:\Users\Monica\Desktop\Projects\Python Projects 1\
23)End_To_End_Projects\Project_3_ML_Scrape_To_Deployment_EndToEnd_SalaryPrediction\
Project_ML_SalaryPrediction

In Anconda Prompt, pip install -r requirements.txt to install all packages.
In Anconda Prompt, write streamlit run salary_prediction.py and press Enter.
Paste URL to browser to check whether working locally or not.
Please be careful with spellings or numbers while typing filename and easier is just copy filename and then run it to avoid any silly errors.
Note: cd <PATH>
[Go to Folder where file is. Select the path from top and right-click and select copy option and paste it next to cd one space <path> and press enter, then you can access all files of that folder]
[cd means change directory]

## Directory Tree/Structure of Project

Folder: 23)End_To_End_Projects >
Project_3_ML_Scrape_To_Deployment_EndToEnd_SalaryPrediction> Project_ML_SalaryPrediction
model_building.ipynb
data_eda.ipynb
data_cleaning.py
data_collection.py
glassdoor_scraper.py
salary_prediction.py

chromdriver.exe [make sure it matches the version of your chrome]

eda_data.csv

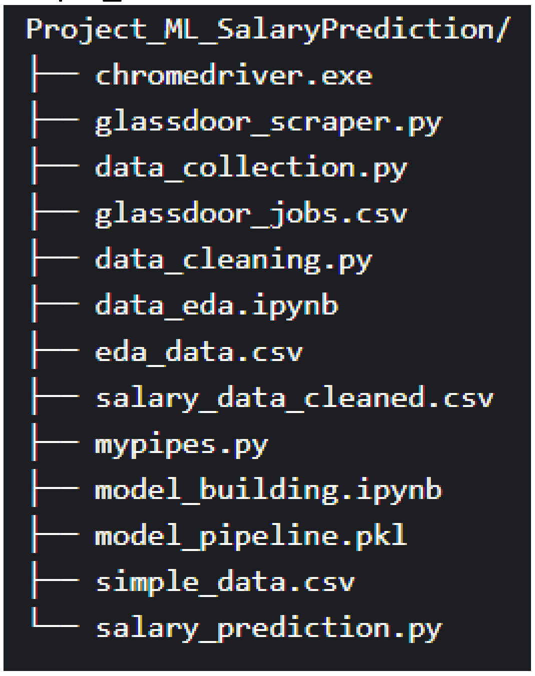
glassdoor_jobs.csv

model_pipeline.pkl

mypipes.py

salary_data_cleaned.csv

simple_data.csv



```
Project_ML_SalaryPrediction/
├── chromedriver.exe
├── glassdoor_scraper.py
├── data_collection.py
├── glassdoor_jobs.csv
├── data_cleaning.py
├── data_eda.ipynb
├── eda_data.csv
├── salary_data_cleaned.csv
├── mypipes.py
├── model_building.ipynb
├── model_pipeline.pkl
├── simple_data.csv
└── salary_prediction.py
```

## To Do/Future Scope

Can make more customize search.

## Technologies Used/System Requirements/Tech Stack

pandas python matplotlib seaborn

scikit learn NumPy Streamlit HEROKU

## Download the Material

You can Download Entire Project from here: [https://github.com/monicadesAI-tech/Project_67](https://github.com/monicadesAI-tech/Project_67)

You can Download Model Building from here: [https://github.com/monicadesAI-tech/Project_67/blob/main/model_building.ipynb](https://github.com/monicadesAI-tech/Project_67/blob/main/model_building.ipynb)

You can Download Web App_File from here: [https://github.com/monicadesAI-tech/Project_67/blob/main/salary_prediction.py](https://github.com/monicadesAI-tech/Project_67/blob/main/salary_prediction.py)

You can see Detailed Project at Website here: [https://github.com/monicadesAI-tech.github.io/project/salarypred.html](https://github.com/monicadesAI-tech.github.io/project/salarypred.html)

Download dependencies from here: [https://github.com/monicadesAI-tech/Project_67/blob/main/requirements.txt](https://github.com/monicadesAI-tech/Project_67/blob/main/requirements.txt)

Download saved model from here: [https://github.com/monicadesAI-tech/Project_67/blob/main/model_pipeline.pkl](https://github.com/monicadesAI-tech/Project_67/blob/main/model_pipeline.pkl)

## Conclusion

- Modelling

Can also apply random forest regression to check if any larger improvement.

- Analysis

Created pipeline and then built model which makes it easy for any future re-use.

## Credits

Ken Jee Channel

## Paper Citation

https://www.ijcseonline.org/spl_pub_paper/IJCSE-ICACICT-2019-16.pdf