Syriatel Take Home Assignment

Tasneem Hosen

14/10/2021

Table of Contents

Figures Table

Tables

Introduction

This report has been prepared in order to submit the ‘ Syriatel Take Home Assignment ’ assignment for AI specialist position.

The applicant:

* Tasneem Hosen
* tasnemhosen91@gmail.com
* +963982052471.

The objective:

Is to use the available information in the job posting to try to predict the salary for the position.

Data Exploration & preparation

1. Reading and discovering the dataset

1.1 Dataset Documentation

dataset name: jobs_train.csv, you can download it here

Our dataset contains 9 main columns and 900,000 record

First 5 rows

	jobId	companyId	jobType	degree	major	industry	yearsExperience	milesFromMetropolis	salary
0	JOB1362685006848	COMP14	MANAGER	MASTERS	MATH	FINANCE	24	36	158
1	JOB1362685403468	COMP36	JANITOR	NONE	NONE	FINANCE	3	4	73
2	JOB1362684563560	COMP56	MANAGER	HIGH_SCHOOL	NONE	SERVICE	11	23	103
3	JOB1362684814664	COMP28	JANITOR	NONE	NONE	EDUCATION	22	42	63
4	JOB1362685123815	COMP48	CEO	DOCTORAL	NONE	OIL	15	66	129

Fig 1: dataset snapshot

We can notice that in our dataset we have:

- Numeric variables: yearsExperience, milesFromMetropolis, salary
- Categorical variables: jobId, companyId, jobType, degree, major, industry
- Target: salary

```
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   jobId                900000 non-null  object
 1   companyId            900000 non-null  object
 2   jobType              900000 non-null  object
 3   degree               900000 non-null  object
 4   major                900000 non-null  object
 5   industry             900000 non-null  object
 6   yearsExperience      900000 non-null  int64
 7   milesFromMetropolis  900000 non-null  int64
 8   salary               900000 non-null  int64
dtypes: int64(3), object(6)
```

Fig 2: dataset info

1.2 Discover categorical variables unique values

jobType:

['MANAGER', 'JANITOR', 'CEO', 'CTO', 'JUNIOR', 'CFO', 'VICE_PRESIDENT', 'SENIOR']

degree:

['MASTERS', 'NONE', 'HIGH_SCHOOL', 'DOCTORAL', 'BACHELORS']

major:

['MATH', 'NONE', 'BIOLOGY', 'COMPSCI', 'CHEMISTRY', 'LITERATURE', 'BUSINESS', 'PHYSICS', 'ENGINEERING']

industry :

['FINANCE', 'SERVICE', 'EDUCATION', 'OIL', 'WEB', 'AUTO', 'HEALTH']

jobId:

all are unique, we are not going to use it as a feature.

companyId:

Has 63 unique values

1.3 Discover the min and max values of our target

The minimum salary value is: 0

The maximum salary value is: 301

We find that only 5 records are with 0 value for salary

	jobId	companyId	jobType	degree	major	industry	yearsExperience	milesFromMetropolis	salary
17780	JOB1362684903671	COMP34	JUNIOR	NONE	NONE	OIL	1	25	0
331320	JOB1362685235843	COMP40	VICE_PRESIDENT	MASTERS	ENGINEERING	WEB	3	29	0
362948	JOB1362685223816	COMP42	MANAGER	DOCTORAL	ENGINEERING	FINANCE	18	6	0
537203	JOB1362685059763	COMP25	CTO	HIGH_SCHOOL	NONE	AUTO	6	60	0
807172	JOB1362684438246	COMP44	JUNIOR	DOCTORAL	MATH	AUTO	11	7	0

Fig 3: dataset records with 0 salary

2. Data Preparation

2.1 Drop all records that have salary equal to 0

```
Int64Index: 899995 entries, 0 to 899999
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   jobId                899995 non-null  object
 1   companyId            899995 non-null  object
 2   jobType              899995 non-null  object
 3   degree               899995 non-null  object
 4   major                899995 non-null  object
 5   industry             899995 non-null  object
 6   yearsExperience      899995 non-null  int64
 7   milesFromMetropolis  899995 non-null  int64
 8   salary               899995 non-null  int64
```

Fig 4: dataset info after dropping records with 0 salary

2.2 Convert non numerical values to categorical ones

```
Int64Index: 899995 entries, 0 to 899999
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   jobId                899995 non-null  object
 1   companyId            899995 non-null  category
 2   jobType              899995 non-null  category
 3   degree               899995 non-null  category
 4   major                899995 non-null  category
 5   industry             899995 non-null  category
 6   yearsExperience      899995 non-null  int64
 7   milesFromMetropolis  899995 non-null  int64
 8   salary               899995 non-null  int64
```

Fig 5: dataset info after data type converting

2.3 Drop unnecessary columns like jobId

3. Features Analyze and Visualization

3.1 Salary Distribution

We are going to plot density plot and boxplot in order to discover our target distribution

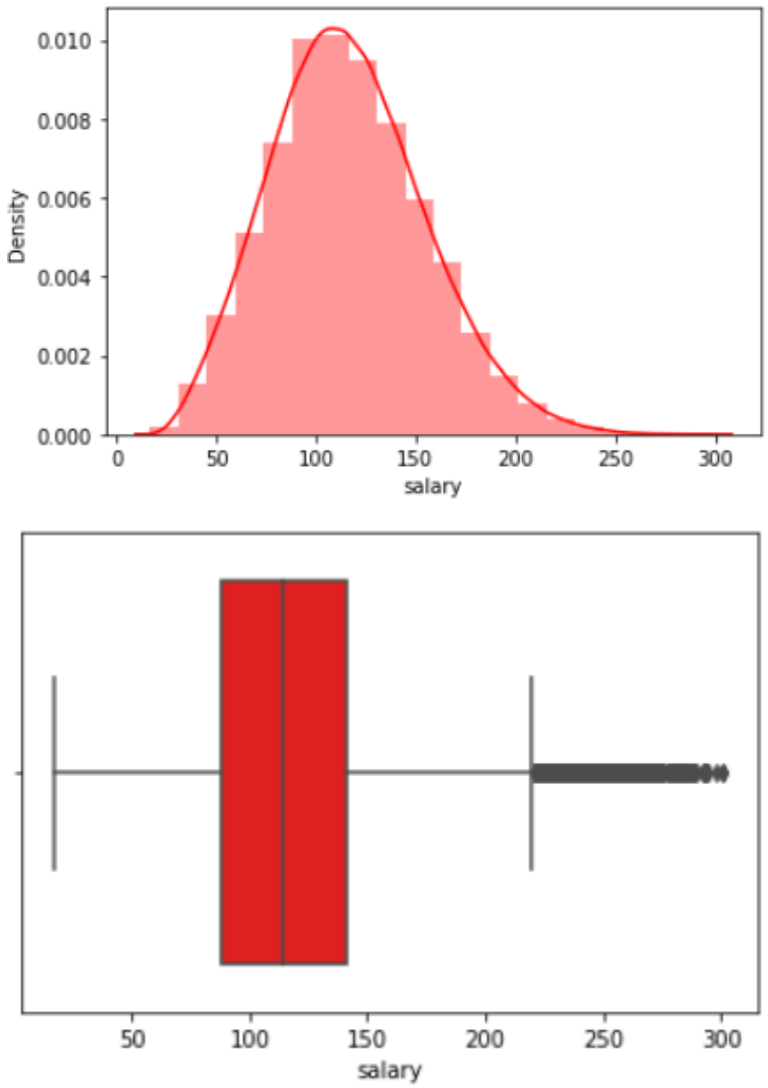


Fig 6: a. salary density plot

b. salary box plot

Note:

- Salary values are a bit equale distrebution between 100 and 150
- Salary values are a bit outlier for values less than 50 and more than 200

3.2 Fetures Distribution

For each categorical feature we are going to plot the feature distribution in two cases:

a. Feature distribution among the dataset (pie plot)
b. Feature distribution with our target (salary) (boxplot)

jobType

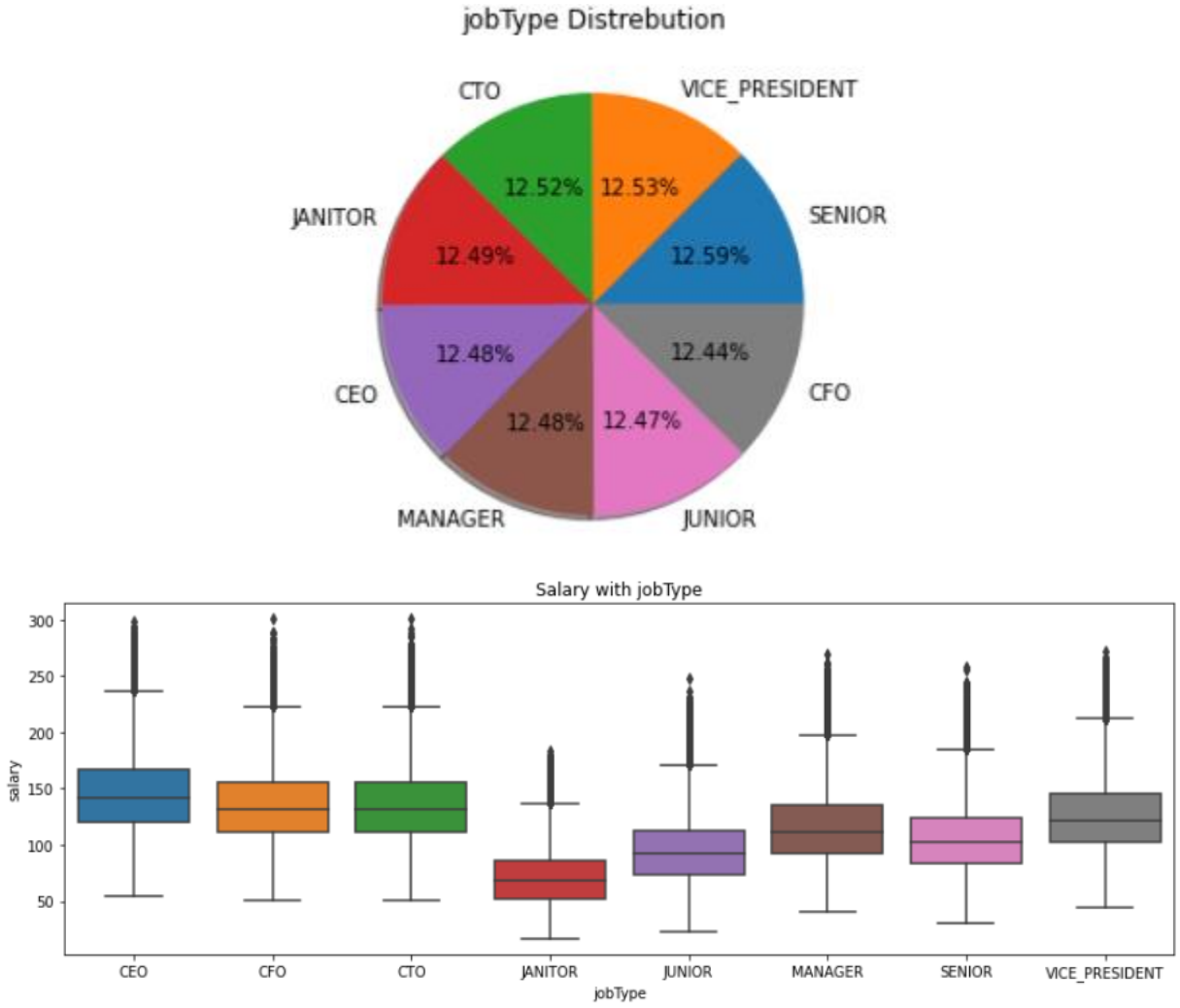


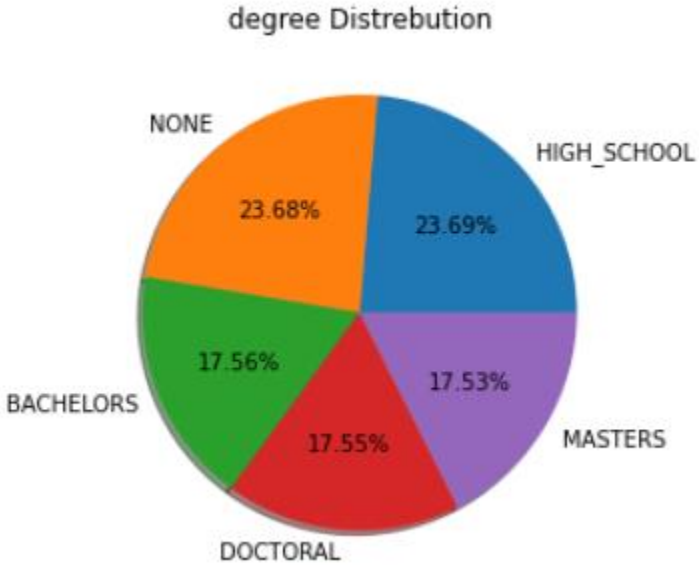
Fig 7: a. jopType pie plot

b. jopType & salary box plot

Note:

- SENIORs: represent the biggest slice in our dataset 12.59 %
- CFOs: represent the smallest slice in our dataset 12.44 %
- CEOs: take the highest salaries
- JANITORs: take the lowest salaries

degree



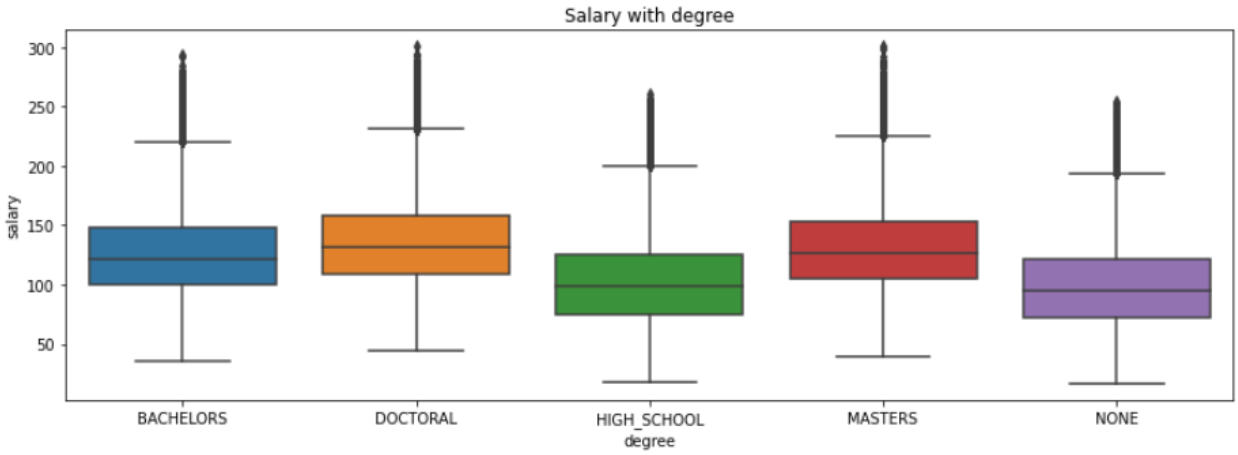


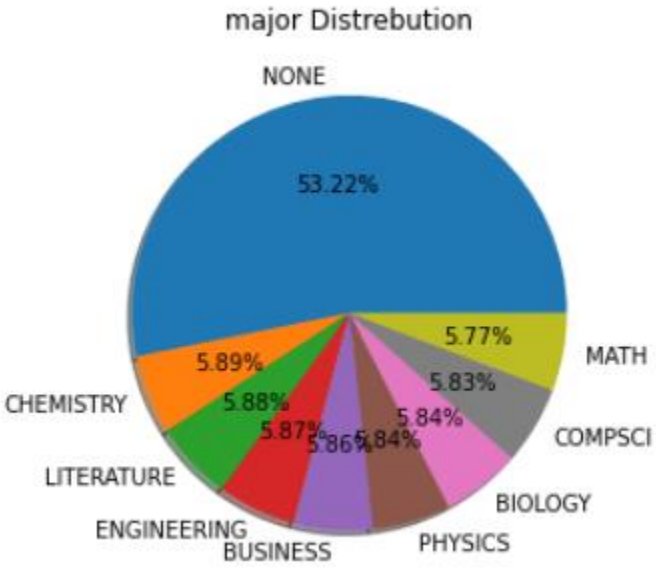
Fig 8: a. degree pie plot

b. degree & salary box plot

Note:

- HIGH_SCHOOL: represents the biggest slice in our dataset 23.69 %
- MASTERS: represents the smallest slice in our dataset 17.53 %
- DOCTORAL: takes the highest salaries
- NONE: takes the lowest salaries

major



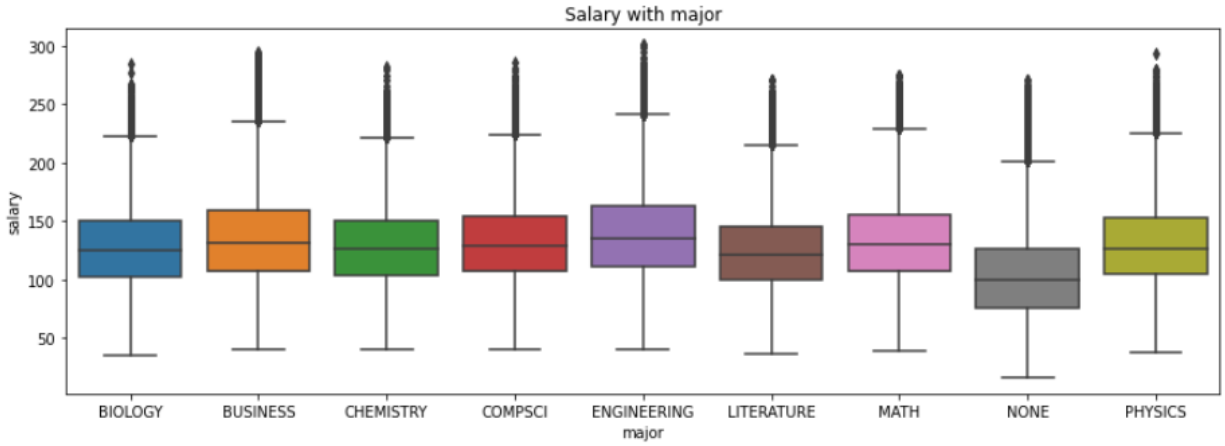


Fig 9: a. major pie plot

b. major & salary box plot

Note:

- NONE: represents the biggest slice in our dataset 23.69 %
- MATH: represents the smallest slice in our dataset 17.53 %
- BUSINESS and ENGINEERING: take the highest salaries
- NONE: takes the lowest salaries

industry

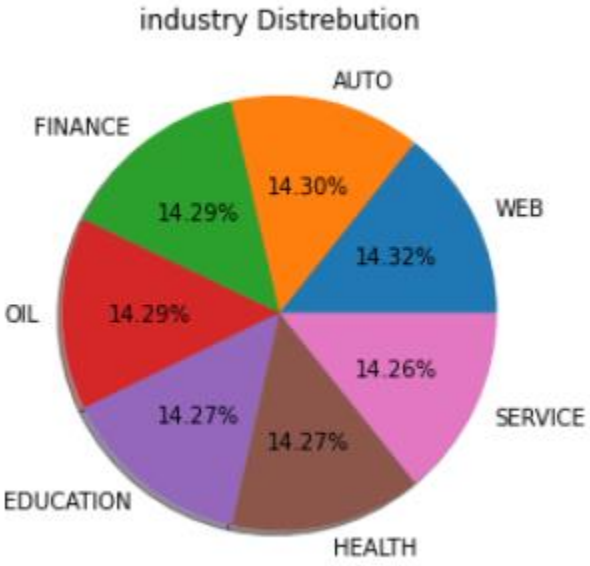


Fig 10: a. industry pie plot

b. industry & salary box plot

Note:

- WEB: represents the biggest slice in our dataset 23.69 %
- SERVICE: represents the smallest slice in our dataset 17.53 %
- FINANCE and OIL: take the highest salaries
- EDUCATION and SERVICE: take the lowest salaries

For each numerical feature we are going to plot the feature distribution in two cases:

a. Feature distribution among the dataset (density plot)
b. Feature distribution with our target (salary) (line plot)

yearsExperience

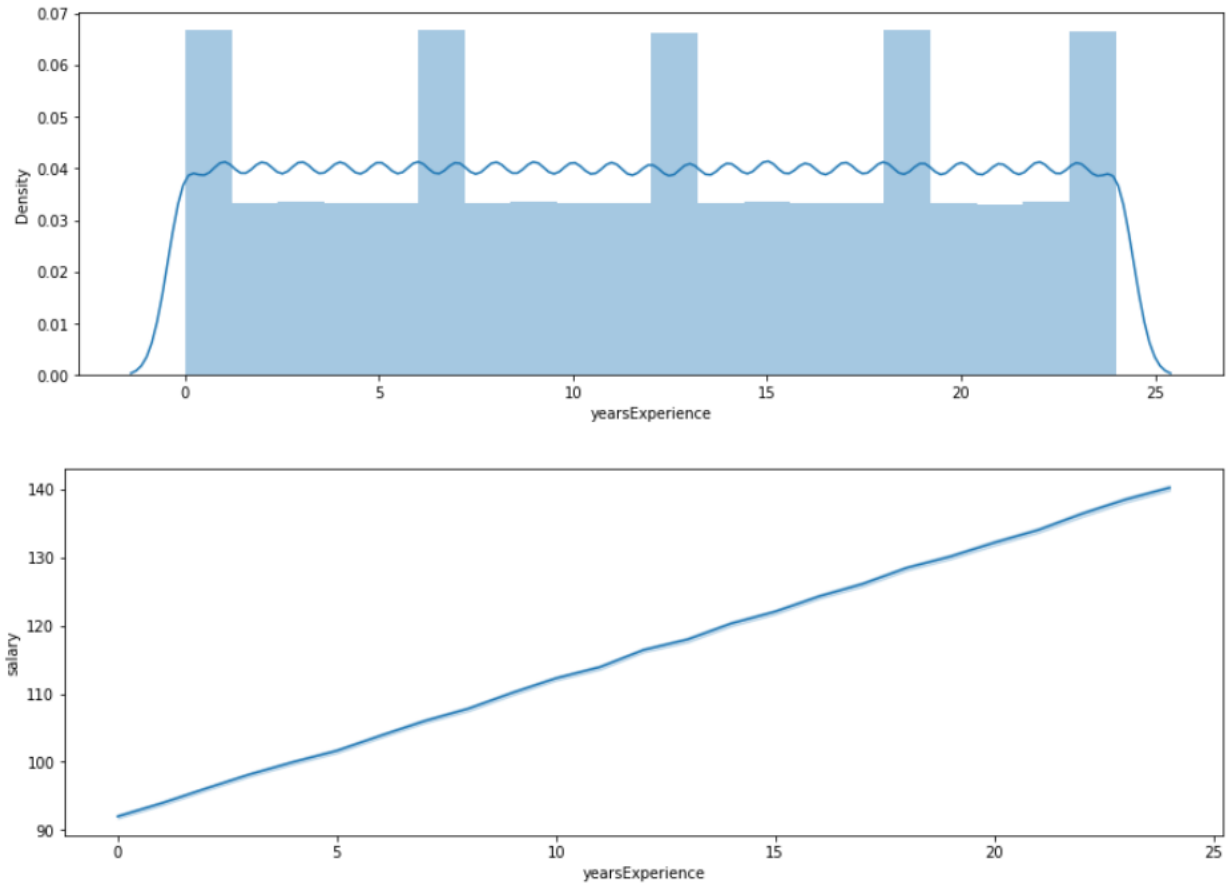


Fig 11: a. yearsExperience density plot

b. yearsExperience & salary line plot

Note:

- Years of experience: 0 and near to 25
- Positive correlation between salary and years of experience

milesFromMetropolis

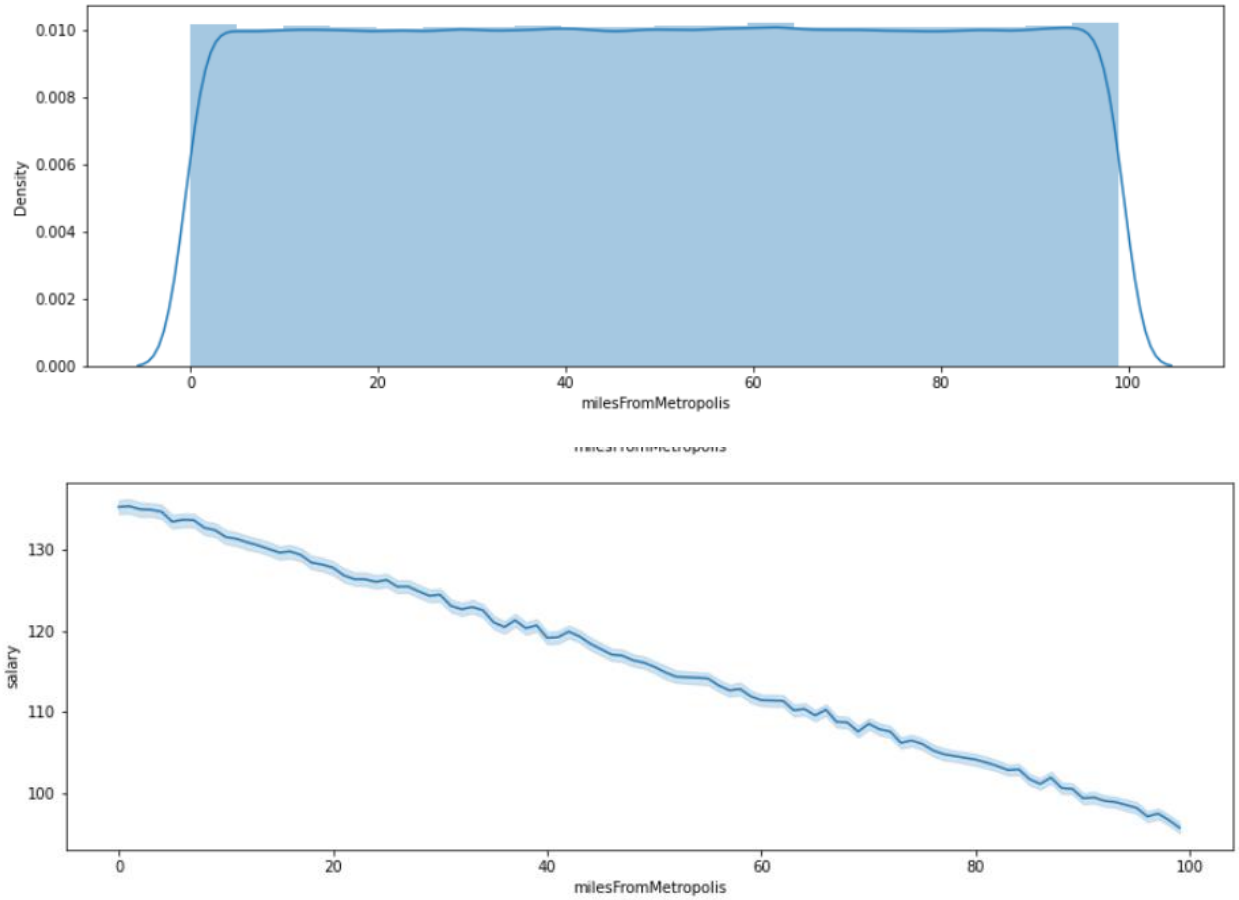


Fig 12: a. milesFromMetropolis density plot

b. milesFromMetropolis & salary line plot

Note:

- Miles From Metropolis: between 0 and near to 100
- Negative correlation between salary and miles From metropolis

3.3 Study features correlation

Finally we are going to plot heat map to discover the correlation between salary and the features, but first we need to encode categorical data to numerical one

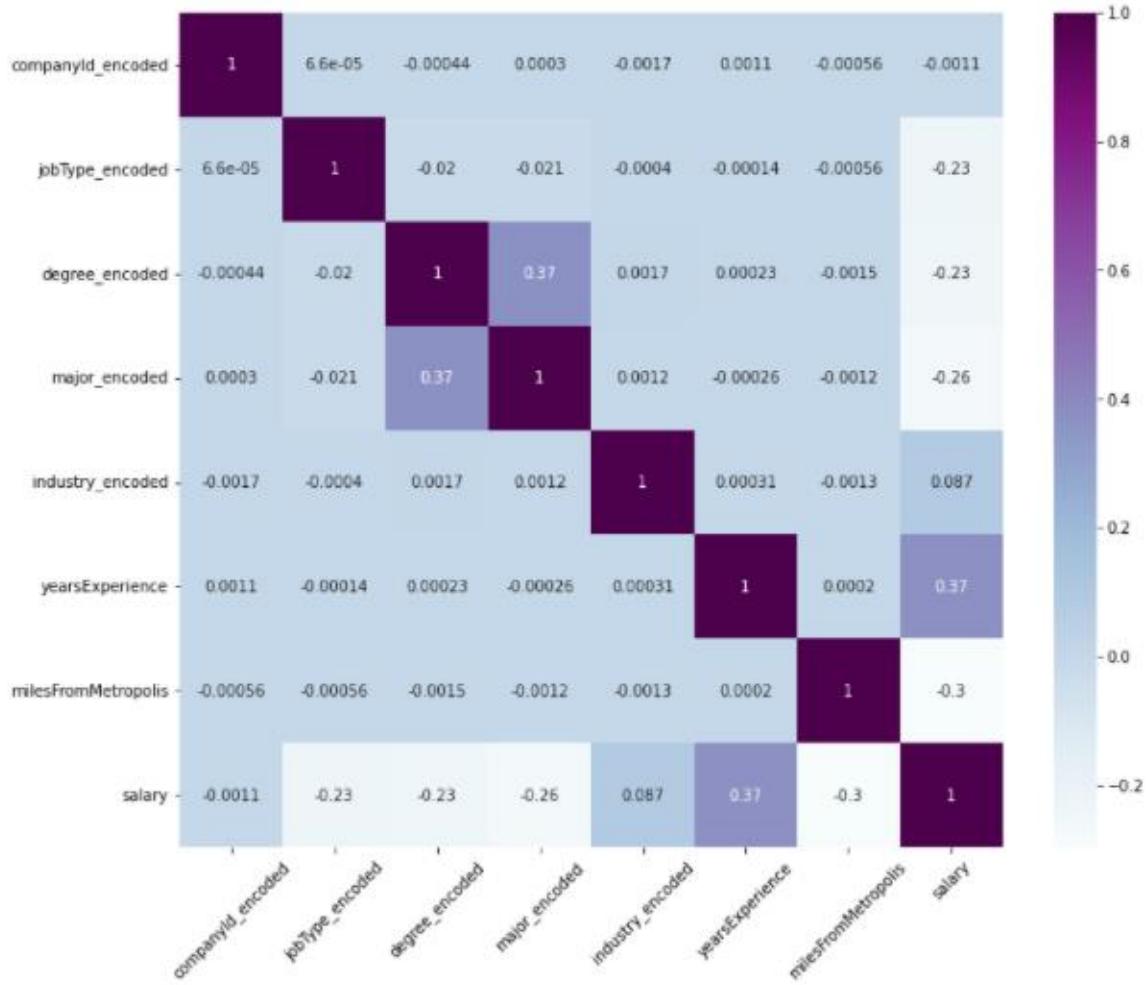


Fig 13: salary/ features heat map

Note:

- Positive correlation between salary and years' experience
- Negative correlation between salary and miles from metropolis

4. Fit the dataset to match the learning algorithm needs then split it

4.1 Apply OneHotEncoding on categorical features

	yearsExperience	milesFromMetropolis	salary	companyId_COMP0	companyId_COMP1	companyId_COMP10	companyId_COMP11	companyId_COMP12
0	24	36	158	0	0	0	0	0
1	3	4	73	0	0	0	0	0
2	11	23	103	0	0	0	0	0
3	22	42	63	0	0	0	0	0
4	15	66	129	0	0	0	0	0
...	...	...	...	...	...	...	...	...
899995	21	40	127	0	0	0	0	0
899996	20	24	129	0	0	0	0	0
899997	21	48	127	0	0	0	0	0
899998	5	78	46	0	0	0	0	0
899999	22	64	119	0	0	0	0	0

Fig 14: dataset after onehotencoding

4.2 Split the dataset into trainig and test sets

Modeling

I applied the following regression algoritms and print their MSE to find the best one:

1. LinearRegression
2. SGDRegressor
3. GradientBoostingRegressor
4. DecisionTreeRegressor
5. RandomForestRegressor

The results where:

Algorithm	MSE
LinearRegression	385.36
SGDRegressor	9658.03
GradientBoostingRegressor	375.33
DecisionTreeRegressor	700.48
RandomForestRegressor	448.14

Table 1: Models' MSE

Key findings

After the final check I used GradientBoostingRegressor where it is an iterative Optimization algorithm used to find the minimum value for a function (usually loss function).

According the result it is necessary to apply one of the optimization algorithm to achive less MSE

Next Step

The most important next steps are:

1- make all the data analyze, preprocessing and visualization processes as functions, and create a python module for each step to make our code professional, clean, and reusable.
2- create a python package contains these modules
3- extract a requirement.txt file which will contain all the needed packages in order to share this code easily
4- In this solution I did not apply any scalar technique, any outlier processing, any hyperparameter tuning. We need to apply them to achieve more accuracy and choose best hyperparameters.

Conclusion

In order to solve this problem I used:

- Python 3.7, jupyter notebook
- When I faced some problems with memory allocate during training the model I used to solve the problem with work on google colab notebook
- Some internet search