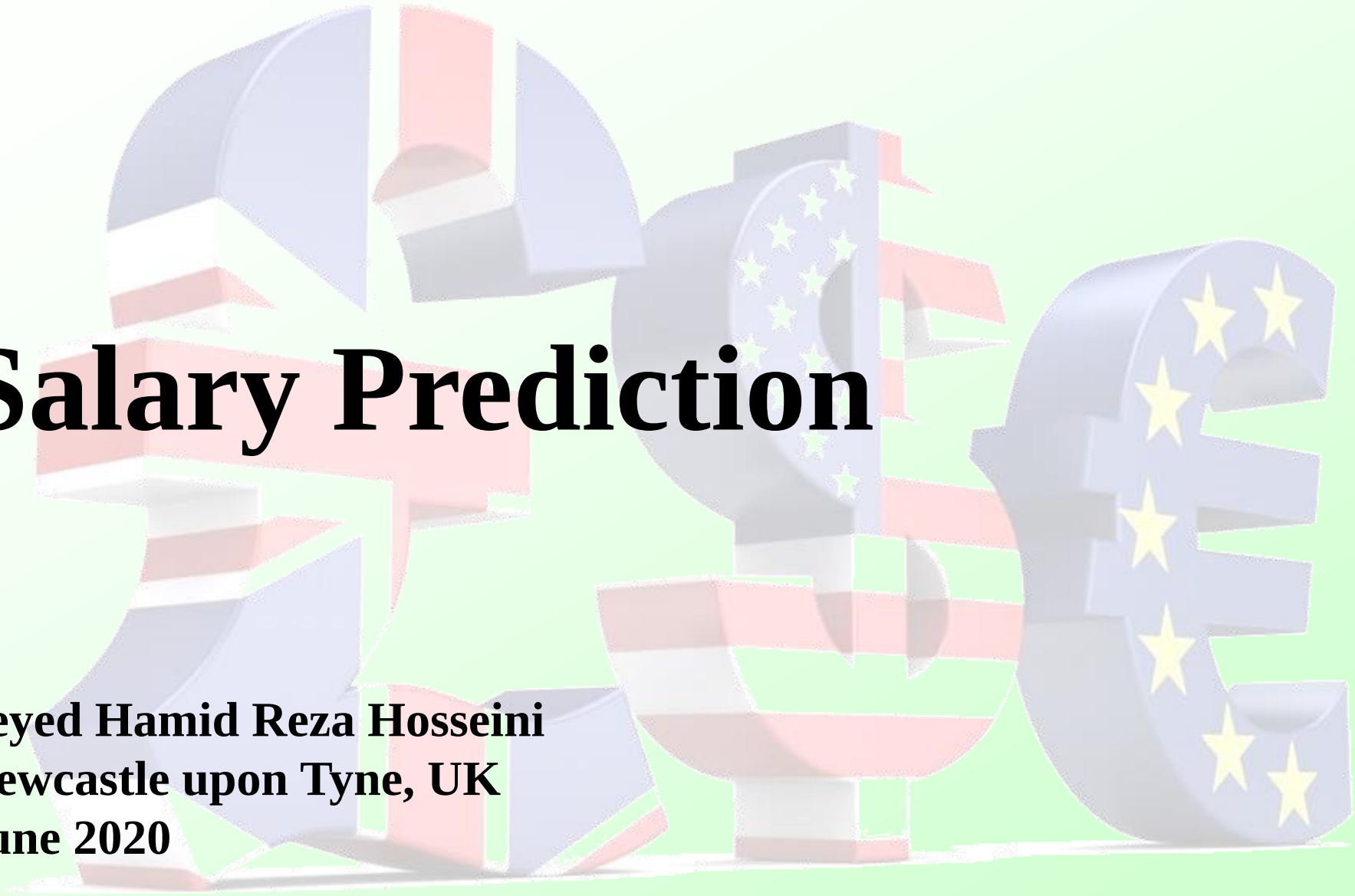


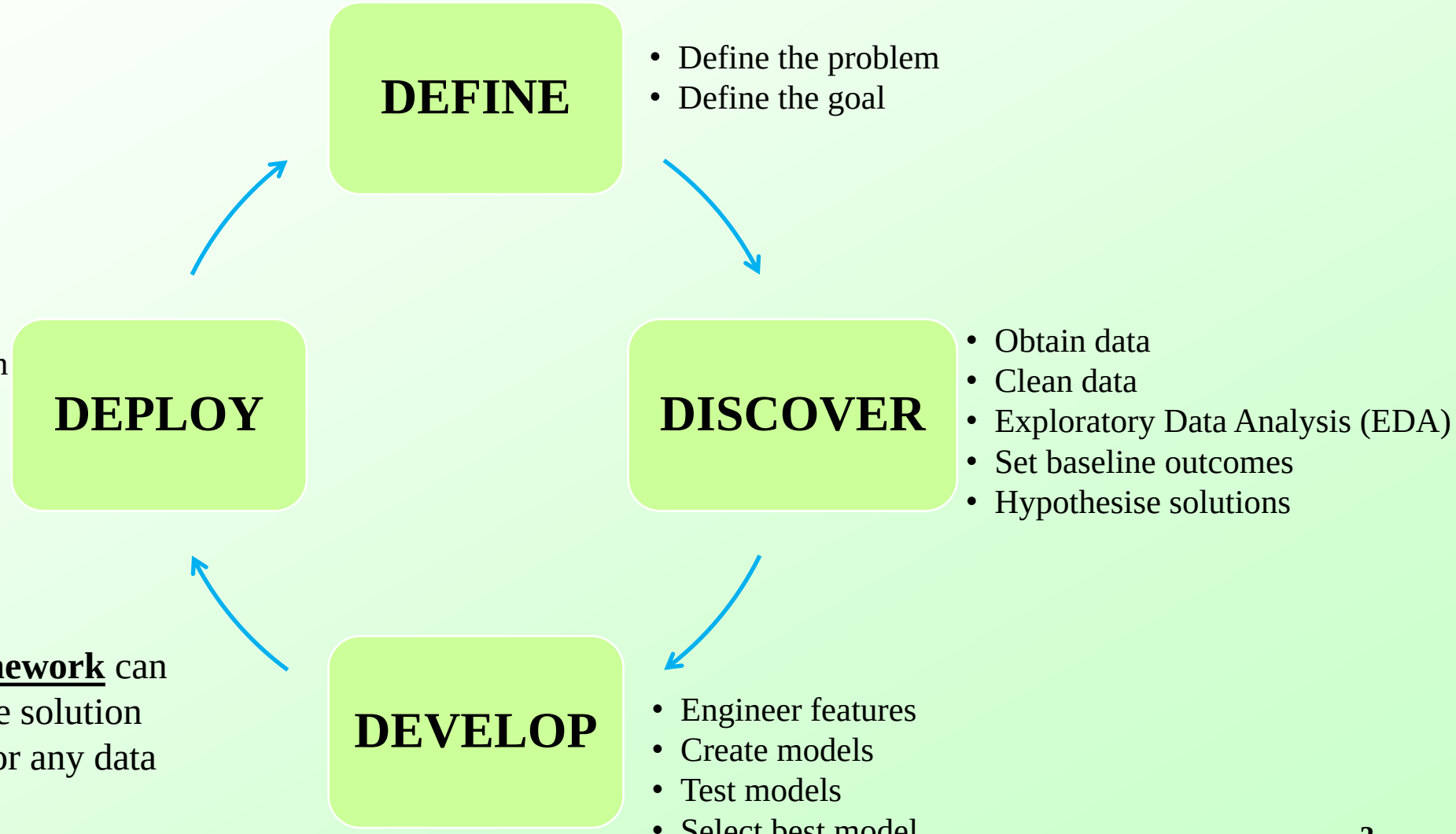
Salary Prediction

Seyed Hamid Reza Hosseini
Newcastle upon Tyne, UK
June 2020

Outline

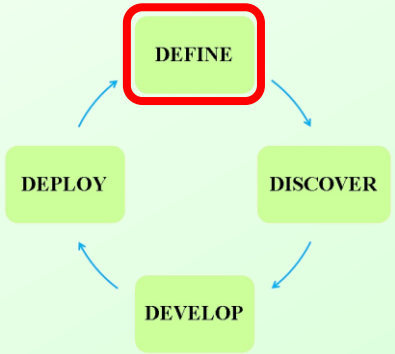
- Solution strategy framework for a data science problem
- Define the problem and goal
- Discover the data
- Develop models
- Deploy the best model
- Next steps

Solution strategy framework for a data science problem



- This generic **4D framework** can be implemented as the solution strategy framework for any data science problem.

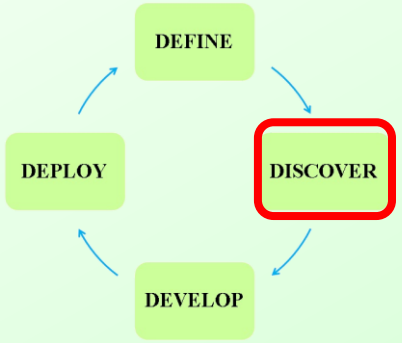
Define the problem and goal



- **The problem**
  The HR manager of a company wants to assign salaries to different posts within the company at the time of publicly advertising the posts. It is crucial that the salaries are within acceptable ranges for the benefit of the company so that the resources of company are not wasted.

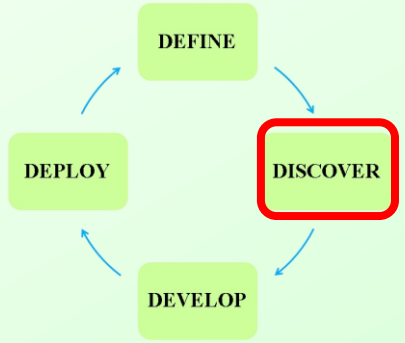
- **The goal**
  The HR manager needs a model which is developed based on an available dataset to help them predict the salary of every post, in order to spend the resources of the company efficiently and create revenue for the company in the long term.

Discover the data (through EDA)



- This is performed through Exploratory Data Analysis (EDA).
- Step 1: Examining and high level overviewing the data
  - Looking at the first e.g. 10 rows of the dataset
  - Purpose:
    - What is the data look like?
    - What features we have got?
    - How the goal of the project can be met?
    - How the dataset needs to be treated to meet the goal?

Discover the data (through EDA)

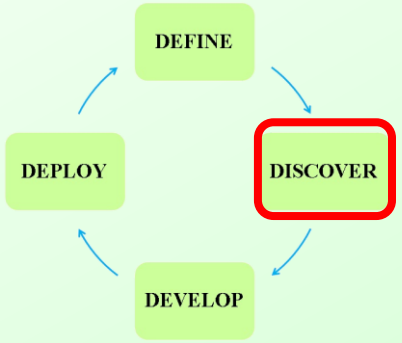


- **Step 1: Examining and high level overviewing the data**

	jobId	companyId	jobType	degree	major	industry	yearsExperience	milesFromMetropolis
0	JOB1362684407687	COMP37	CFO	MASTERS	MATH	HEALTH	10	83
1	JOB1362684407688	COMP19	CEO	HIGH_SCHOOL	NONE	WEB	3	73
2	JOB1362684407689	COMP52	VICE_PRESIDENT	DOCTORAL	PHYSICS	HEALTH	10	38
3	JOB1362684407690	COMP38	MANAGER	DOCTORAL	CHEMISTRY	AUTO	8	17
4	JOB1362684407691	COMP7	VICE_PRESIDENT	BACHELORS	PHYSICS	FINANCE	8	16
5	JOB1362684407692	COMP15	MANAGER	DOCTORAL	COMPSCI	FINANCE	2	31
6	JOB1362684407693	COMP15	CFO	NONE	NONE	HEALTH	23	24
7	JOB1362684407694	COMP24	JUNIOR	BACHELORS	CHEMISTRY	EDUCATION	9	70
8	JOB1362684407695	COMP20	JANITOR	HIGH_SCHOOL	NONE	EDUCATION	1	54
9	JOB1362684407696	COMP41	VICE_PRESIDENT	BACHELORS	CHEMISTRY	AUTO	17	68

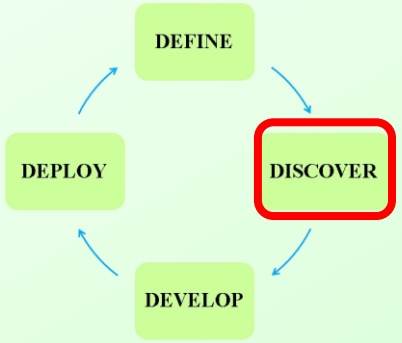
- The column ‘jobId’ seems to be unique.
- The columns ‘companyId’, ‘jobType’, ‘degree’, ‘major’ and ‘industry’ seem to be categorical columns since there are some repetitions. Therefore, the number of unique categories for each column are expected to be much less than the total number of records.

Discover the data (through EDA)



- **Step 2: Inspect more detail of the dataset**
  - **Performed using .info()**
  - **Purpose:**
    - **Understanding some detail information of the dataset:**
      - **Length and columns of the dataset**
      - **Datatypes appeared in the dataset**

Discover the data (through EDA)



- **Step 2: Inspect more detail of the dataset**

- Total number of records: 1,000,000
- Object columns:
  - ‘jobId’
  - ‘companyId’
  - ‘jobType’
  - ‘degree’
  - ‘major’
  - ‘industry’
- Integer columns:
  - ‘yearsExperience’
  - ‘milesFromMetropolis’
  - ‘salary’
- The object columns are categorical.
- The integer columns are numeric.

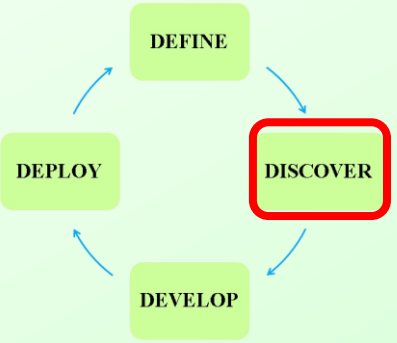
```
train_feature_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000000 entries, 0 to 999999
Data columns (total 8 columns):
jobId                  1000000 non-null object
companyId              1000000 non-null object
jobType                1000000 non-null object
degree                 1000000 non-null object
major                  1000000 non-null object
industry               1000000 non-null object
yearsExperience        1000000 non-null int64
milesFromMetropolis    1000000 non-null int64
dtypes: int64(2), object(6)
memory usage: 61.0+ MB
```

```
train_target_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000000 entries, 0 to 999999
Data columns (total 2 columns):
jobId     1000000 non-null object
salary    1000000 non-null int64
dtypes: int64(1), object(1)
memory usage: 15.3+ MB
```

Discover the data (through EDA)



- **Step 3: Checking for duplicates**
  - **Purpose:**
    - **Are they just because of some duplication?**
    - **Are they as a result of some sort of data corruption/missing?**

- There are not any duplicates.
- In case of presence of any duplicates, this needed to be investigated further. It should be decided how to handle them, whether or not they should be dropped.

```
train_feature_df.duplicated().sum()
```

```
0
```

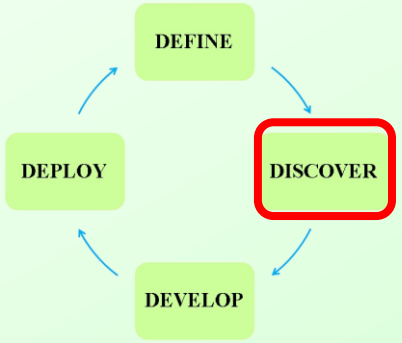
```
train_target_df.duplicated().sum()
```

```
0
```

```
test_feature_df.duplicated().sum()
```

```
0
```

Discover the data (through EDA)



- **Step 4: Identification of numerical and categorical features**
  - **Purpose:**
    - **To treat numerical features different from categorical features**

```
train_feature_df.columns
```

```
Index(['jobId', 'companyId', 'jobType', 'degree', 'major', 'industry',
       'yearsExperience', 'milesFromMetropolis'],
      dtype='object')
```

Numeric columns →

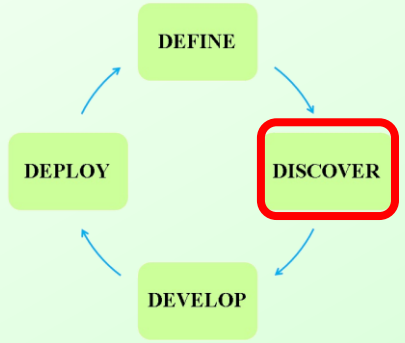
```
numeric_cols = ['yearsExperience', 'milesFromMetropolis']
```

Categorical columns →

```
categorical_cols = ['jobId', 'companyId', 'jobType', 'degree', 'major', 'industry']
```

The column 'salary' in train_target_df is numeric and the target.

Discover the data (through EDA)



- **Step 5: Summarising numerical and categorical variables separately**
  - **Purpose:**
    - **To learn which methods needs to be built-in for each of the variables**

```
train_feature_df.describe(include = [np.number])
```

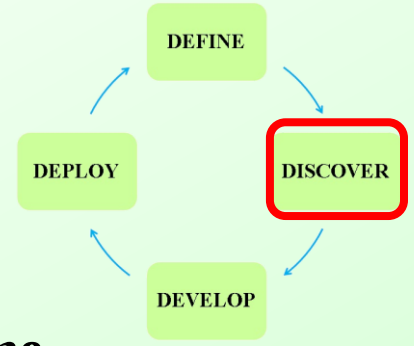
	yearsExperience	milesFromMetropolis
count	1000000.000000	1000000.000000
mean	11.992386	49.529260
std	7.212391	28.877733
min	0.000000	0.000000
25%	6.000000	25.000000
50%	12.000000	50.000000
75%	18.000000	75.000000
max	24.000000	99.000000

```
train_feature_df.describe(include = ['O'])
```

	jobId	companyId	jobType	degree	major	industry
count	1000000	1000000	1000000	1000000	1000000	1000000
unique	1000000	63	8	5	9	7
top	JOB1362685268223	COMP39	SENIOR	HIGH_SCHOOL	NONE	WEB
freq	1	16193	125886	236976	532355	143206

- The numerical columns have reasonable values and ranges, and there is no need to handle any unexpected data.
- 'jobId' is some randomly generated value, and does not impact the target variable (Salary). Hence, it is not a feature to be considered in modelling. Therefore, it will be dropped from the training set later.
- It seems the dataframe is nice and clean.

Discover the data (through EDA)



- **Step 6: Merging features and target of training dataframe into a single dataframe**
  - **Purpose:**
    - **To make the rest of the procedure easier**

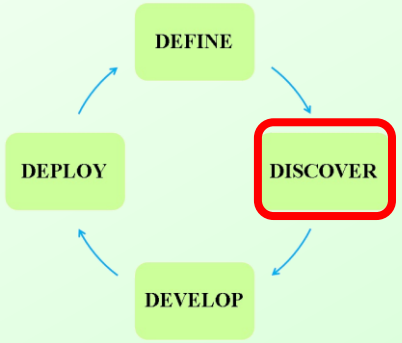
```
# Merge the features and salaries on jobId, delete original file to save memory
train_df = pd.merge(train_feature_df, train_target_df, on='jobId')
del train_feature_df
del train_target_df
```

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000000 entries, 0 to 999999
Data columns (total 9 columns):
jobId                  1000000 non-null object
companyId              1000000 non-null object
jobType                1000000 non-null object
degree                 1000000 non-null object
major                  1000000 non-null object
industry               1000000 non-null object
yearsExperience        1000000 non-null int64
milesFromMetropolis    1000000 non-null int64
salary                 1000000 non-null int64
dtypes: int64(3), object(6)
memory usage: 76.3+ MB
```

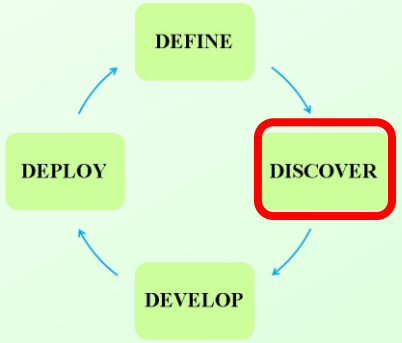
- It is made sure that the training features and targets are merged correctly.
- There are 1,000,000 records.
- The features are followed by the salaries.
- Hence, everything looks acceptable.

Discover the data (through EDA)

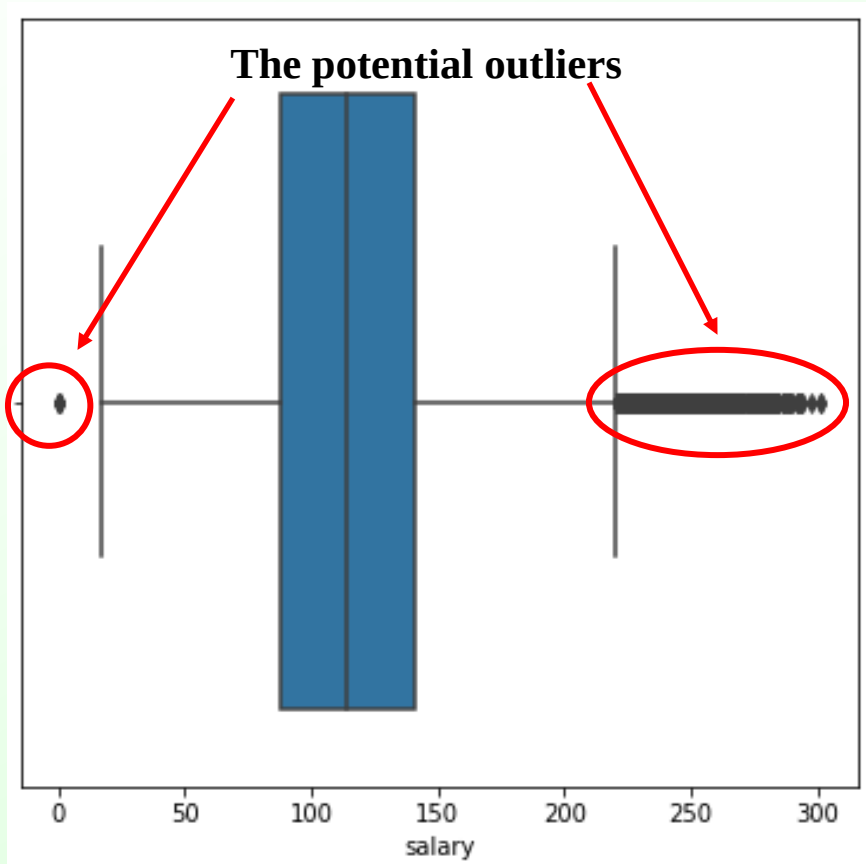


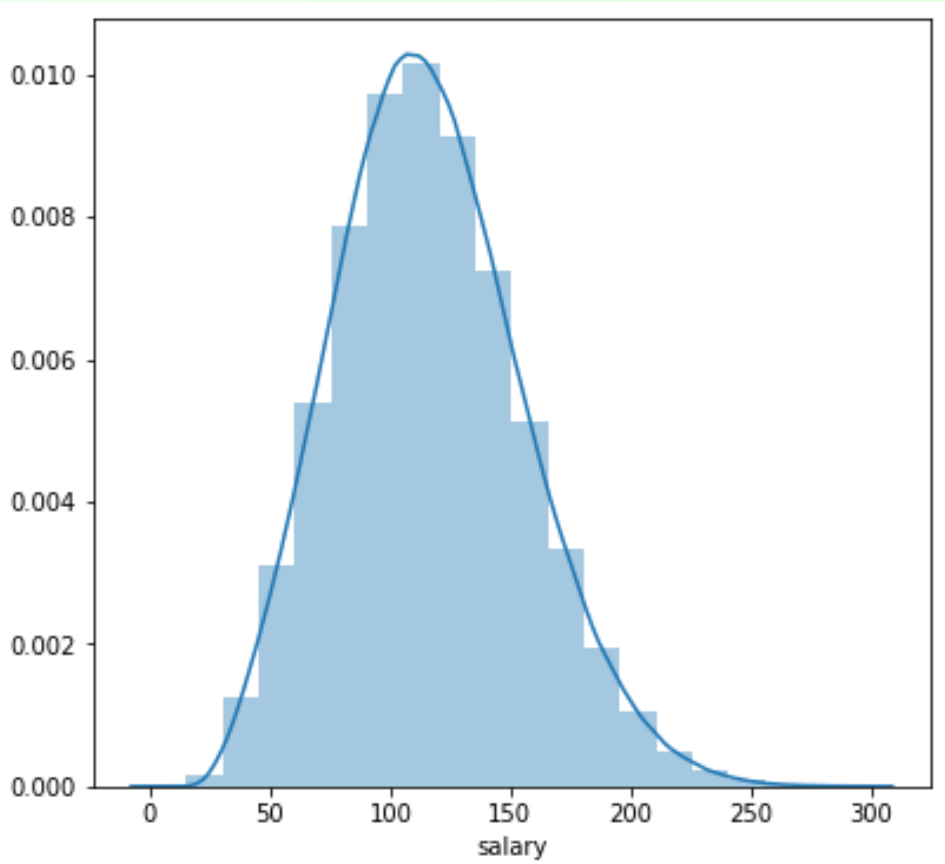
- **Step 7: Visualising the target variable (Salary)**
  - **Purpose:**
    - **To see if there is any pattern in the target variable**
    - **To obtain an idea for further steps to deal with the target variable of the dataframe**

Discover the data (through EDA)



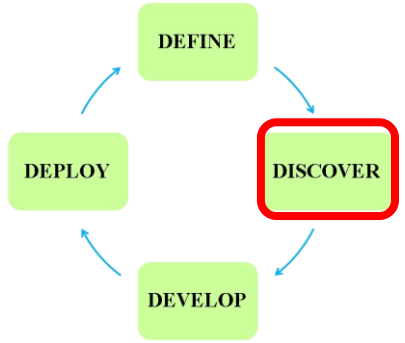
- **Step 7: Visualising the target variable (Salary)**



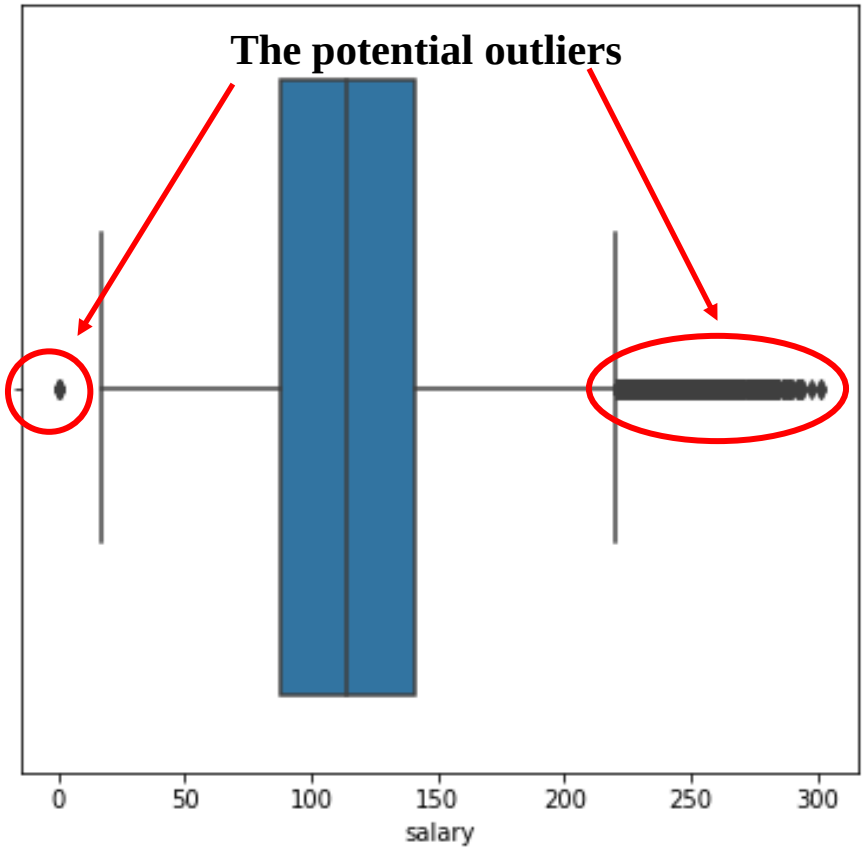


- The potential outliers need further investigation.
- The ‘salary’ follows similar to a Normal distribution, with a mean around 120.

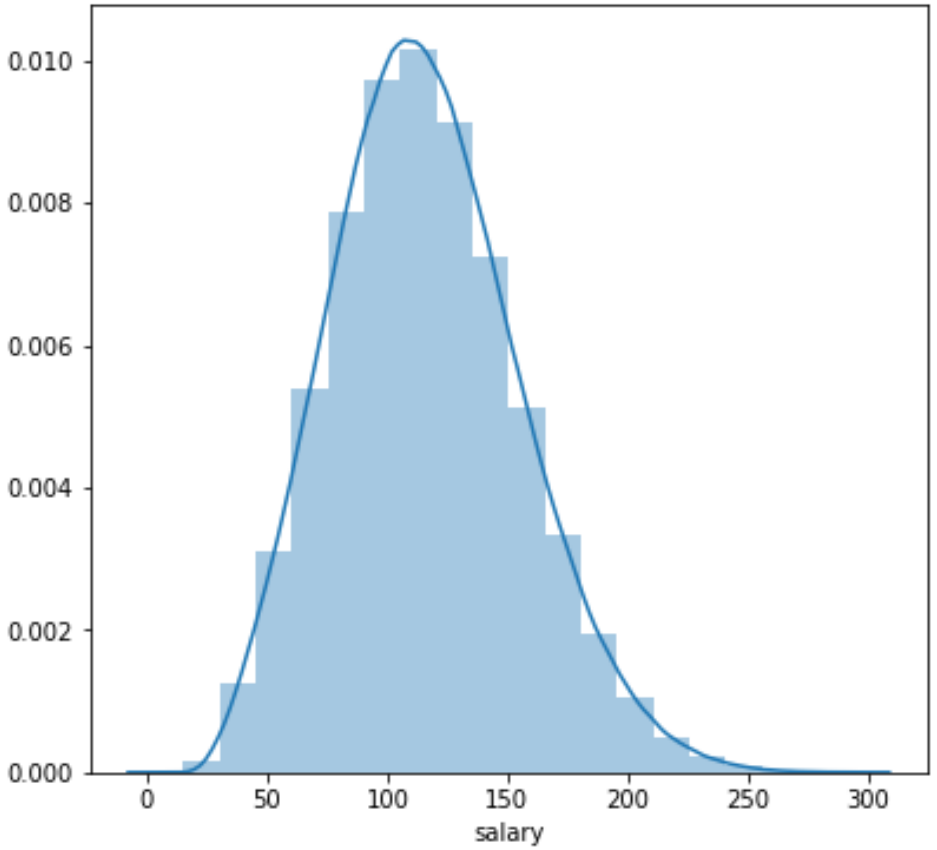
Discover the data (through EDA)



- **Step 7: Visualising the target variable (Salary)**

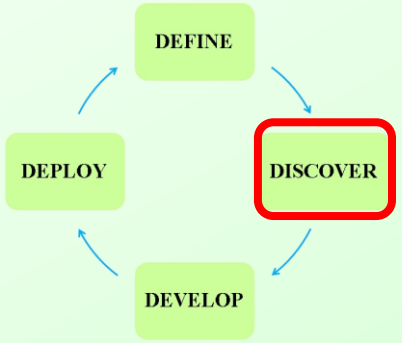


- The potential outliers need further investigation.



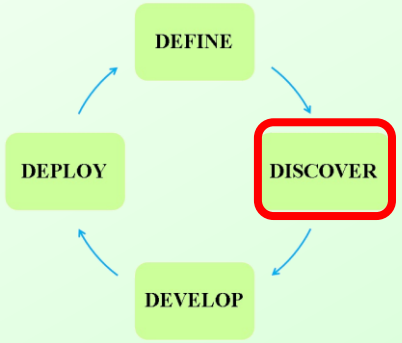
- The ‘salary’ follows similar to a Normal distribution, with a mean around 120.

Discover the data (through EDA)



- **Step 8: Dealing with the outliers**
- **Step 8.1: Identification of potential outliers**
  - **This is performed using Interquartile Range (IQR) rule**
  - **Purpose:**
    - **To define an acceptable range for the data**
    - **To use IQR in the next step to inspect the potential outliers**

Discover the data (through EDA)



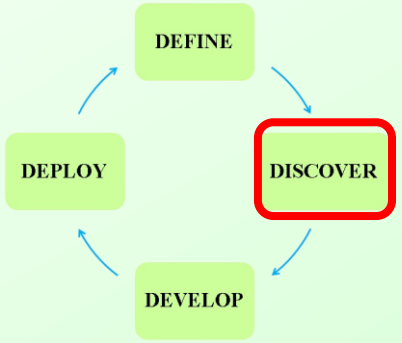
- **Step 8.1: Identification of potential outliers**

```
stat = train_df.salary.describe()
print(stat)
IQR = stat['75%'] - stat['25%']
upper = stat['75%'] + 1.5 * IQR
lower = stat['25%'] - 1.5 * IQR
print('The upper and lower bounds for suspected outliers are {} and {} .'.format(upper, lower))
```

```
count    1000000.000000
mean         116.061818
std           38.717936
min            0.000000
25%           88.000000
50%          114.000000
75%          141.000000
max          301.000000
Name: salary, dtype: float64
The upper and lower bounds for suspected outliers are 220.5 and 8.5 .
```

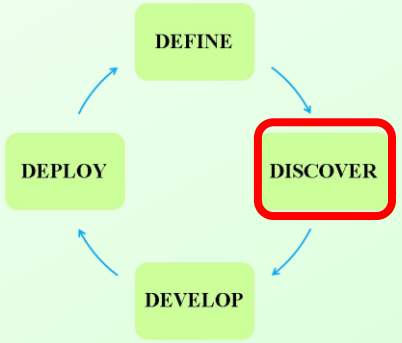
- Hence, any data entry that has the salary below 8.5 or above 220.5 is considered as potential outlier and needs to be further investigated.

Discover the data (through EDA)



- **Step 8.2: Examining potential outliers**
  - **Performed using the IQR range determined in Step 8.1**
  - **Purpose:**
    - **To learn if the outliers are missing and/or meaningful data**
    - **To decide whether to include them in the training set or should they be excluded.**

Discover the data (through EDA)



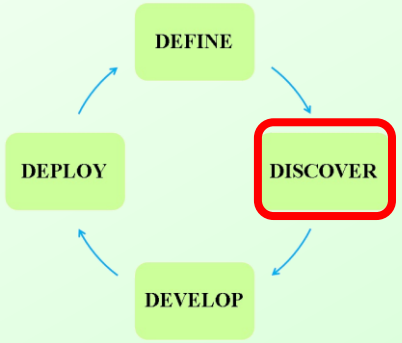
- **Step 8.2: Examining potential outliers**
  - **Inspection of the data entries with salaries below 8.5.**

```
#check potential outlier below lower bound
train_df[train_df.salary < lower]
```

	jobId	companyId	jobType	degree	major	industry	yearsExperience	milesFromMetropolis	salary
30559	JOB1362684438246	COMP44	JUNIOR	DOCTORAL	MATH	AUTO	11	7	0
495984	JOB1362684903671	COMP34	JUNIOR	NONE	NONE	OIL	1	25	0
652076	JOB1362685059763	COMP25	CTO	HIGH_SCHOOL	NONE	AUTO	6	60	0
816129	JOB1362685223816	COMP42	MANAGER	DOCTORAL	ENGINEERING	FINANCE	18	6	0
828156	JOB1362685235843	COMP40	VICE_PRESIDENT	MASTERS	ENGINEERING	WEB	3	29	0

- These entries with zero salary do not appear to be volunteer positions.(e.g. a Manager, with Doctoral degree definitely should have a salary.)
- Hence, they are instances of missing/corrupt data and should be removed from the training set.

Discover the data (through EDA)



- **Step 8.2: Examining potential outliers**
  - **Inspection of the data entries with salaries above 220.5.**

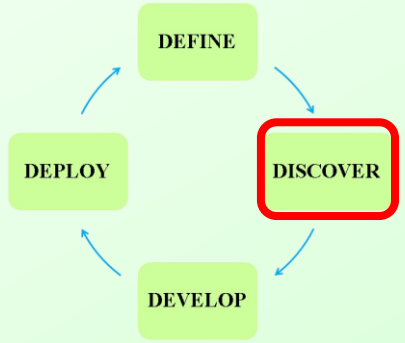
```
#check potential outlier above upper bound
train_df.loc[train_df.salary > upper, 'jobType'].value_counts()
```

```
CEO                3227
CFO                1496
CTO                1488
VICE_PRESIDENT      603
MANAGER             217
SENIOR               66
JUNIOR               20
Name: jobType, dtype: int64
```

These are all senior or higher roles, and it is totally reasonable that they earn lot of money.

However, this is strange that people in junior roles are making lot of money. Hence, these suspicious entries need to be further investigated, which is carried out in slide 20.

Discover the data (through EDA)



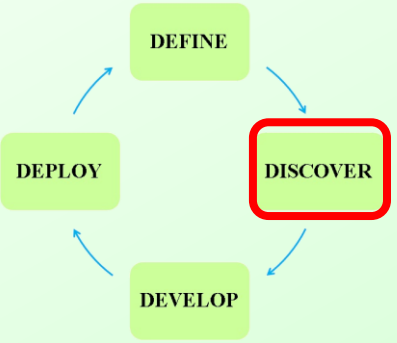
- **Step 8.2: Examining potential outliers**
  - **Inspection of the data entries with salaries above 220.5.**

Inspection of these suspicious entries shows that these are primarily in oil, finance and web industries. Also, most of them have advanced degrees. Hence, these are genuine correct data since these industries have high salaries even though in their Junior roles. Hence, these data are true outliers and will not be dropped from the dataset.

```
# Check most suspicious potential outliers above upper bound, i.e. JUNIOR roles, to make sure if they
# should be excluded from trainig set or not.
train_df[(train_df.salary > upper) & (train_df.jobType == 'JUNIOR')]
```

	jobId	companyId	jobType	degree	major	industry	yearsExperience	milesFromMetropolis	salary
1222	JOB1362684408909	COMP40	JUNIOR	MASTERS	COMPSCI	OIL	24	5	225
27710	JOB1362684435397	COMP21	JUNIOR	DOCTORAL	ENGINEERING	OIL	24	3	246
31355	JOB1362684439042	COMP45	JUNIOR	DOCTORAL	COMPSCI	FINANCE	24	0	225
100042	JOB1362684507729	COMP17	JUNIOR	DOCTORAL	BUSINESS	FINANCE	23	8	248
160333	JOB1362684568020	COMP18	JUNIOR	DOCTORAL	BUSINESS	FINANCE	22	3	223
189582	JOB1362684597269	COMP32	JUNIOR	DOCTORAL	BUSINESS	OIL	24	11	221
214606	JOB1362684622293	COMP47	JUNIOR	MASTERS	BUSINESS	FINANCE	22	4	222
303778	JOB1362684711465	COMP51	JUNIOR	MASTERS	ENGINEERING	WEB	24	2	226
348354	JOB1362684756041	COMP56	JUNIOR	DOCTORAL	ENGINEERING	OIL	23	25	226
427593	JOB1362684835280	COMP54	JUNIOR	DOCTORAL	ENGINEERING	FINANCE	23	3	221
500739	JOB1362684908426	COMP40	JUNIOR	DOCTORAL	ENGINEERING	OIL	21	0	227
627534	JOB1362685035221	COMP5	JUNIOR	DOCTORAL	ENGINEERING	OIL	24	29	230
645555	JOB1362685053242	COMP36	JUNIOR	DOCTORAL	BUSINESS	FINANCE	24	1	225
656572	JOB1362685064259	COMP28	JUNIOR	DOCTORAL	BUSINESS	OIL	22	3	222
685775	JOB1362685093462	COMP38	JUNIOR	BACHELORS	ENGINEERING	OIL	24	13	225
743326	JOB1362685151013	COMP14	JUNIOR	DOCTORAL	BUSINESS	FINANCE	19	0	236
787674	JOB1362685195361	COMP43	JUNIOR	DOCTORAL	BUSINESS	FINANCE	18	15	232
796956	JOB1362685204643	COMP30	JUNIOR	MASTERS	BUSINESS	OIL	24	2	228
855219	JOB1362685262906	COMP13	JUNIOR	MASTERS	ENGINEERING	OIL	22	26	225
954368	JOB1362685362055	COMP11	JUNIOR	DOCTORAL	BUSINESS	OIL	24	26	223

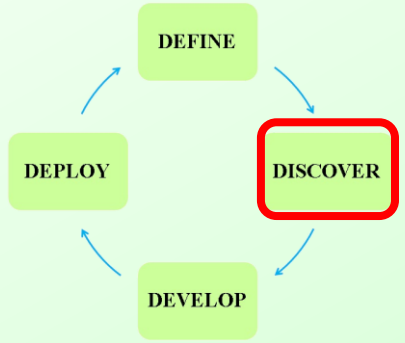
Discover the data (through EDA)



- **Step 8.3: Removing data with zero salaries**
  - **Purpose:**
    - **To clean the data and have a meaningful and legitimate dataset**

```
train_df = train_df[train_df.salary > lower]
```

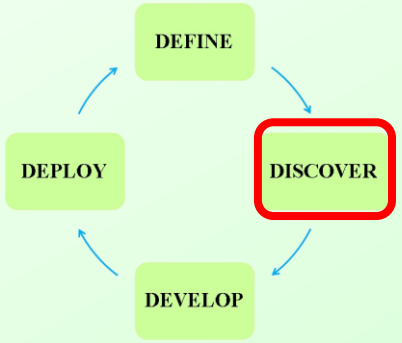
Discover the data (through EDA)



- **Step 9: Plot all the features separately**
  - **A function (plot_feature) is developed for this purpose.**
  - **Purpose:**
    - **To examine the correlation between the target (Salary) and every feature**

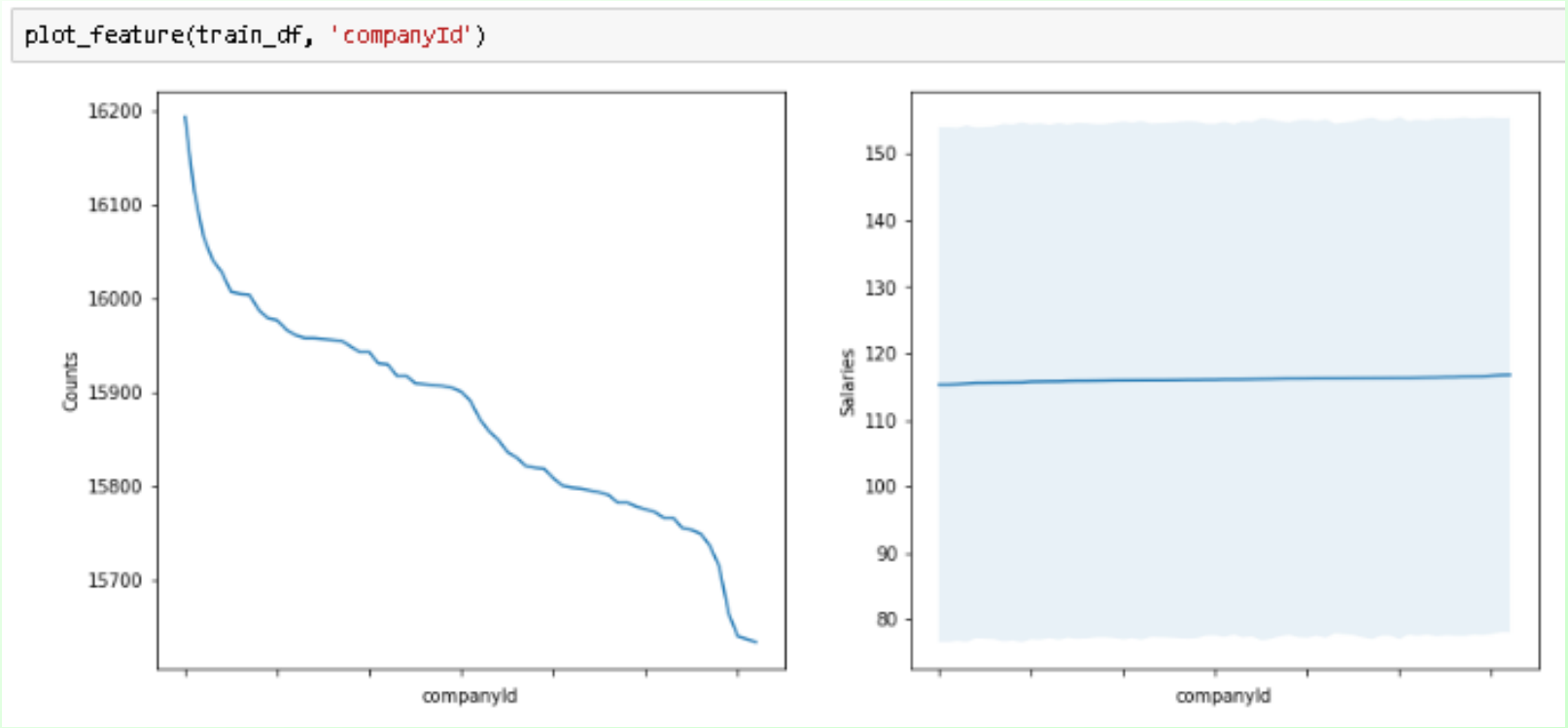
```
def plot_feature(df, col):
    '''
    Make plot for each features
    left, the distribution of samples on the feature
    right, the dependance of salary on the feature
    '''
    plt.figure(figsize = (14, 6))
    plt.subplot(1, 2, 1)
    if df[col].dtype == 'int64':
        df[col].value_counts().sort_index().plot()
    else:
        #change the categorical variable to category type and order their level by the mean salary
        #in each category
        mean = df.groupby(col)['salary'].mean()
        df[col] = df[col].astype('category')
        levels = mean.sort_values().index.tolist()
        df[col].cat.reorder_categories(levels, inplace=True)
        df[col].value_counts().plot()
    plt.xticks(rotation=45)
    plt.xlabel(col)
    plt.ylabel('Counts')
    plt.subplot(1, 2, 2)

    if df[col].dtype == 'int64' or col == 'companyId':
        #plot the mean salary for each category and fill between the (mean - std, mean + std)
        mean = df.groupby(col)['salary'].mean()
        std = df.groupby(col)['salary'].std()
        mean.plot()
        plt.fill_between(range(len(std.index)), mean.values-std.values, mean.values + std.values, \
                         alpha = 0.1)
    else:
        sns.boxplot(x = col, y = 'salary', data=df)

    plt.xticks(rotation=45)
    plt.ylabel('Salaries')
    plt.show()
```

Discover the data (through EDA)



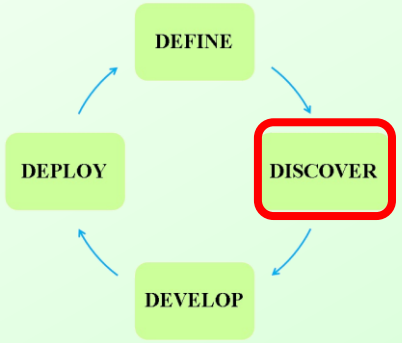
- **Step 9: Plot all the features separately (‘companyId’)**

```
plot_feature(train_df, 'companyId')
```



- The straight line shows that this feature is not predictive of the target, meaning the salaries are very weakly associated with these randomly generated numbers.
- Hence, these will be removed from the training dataset.

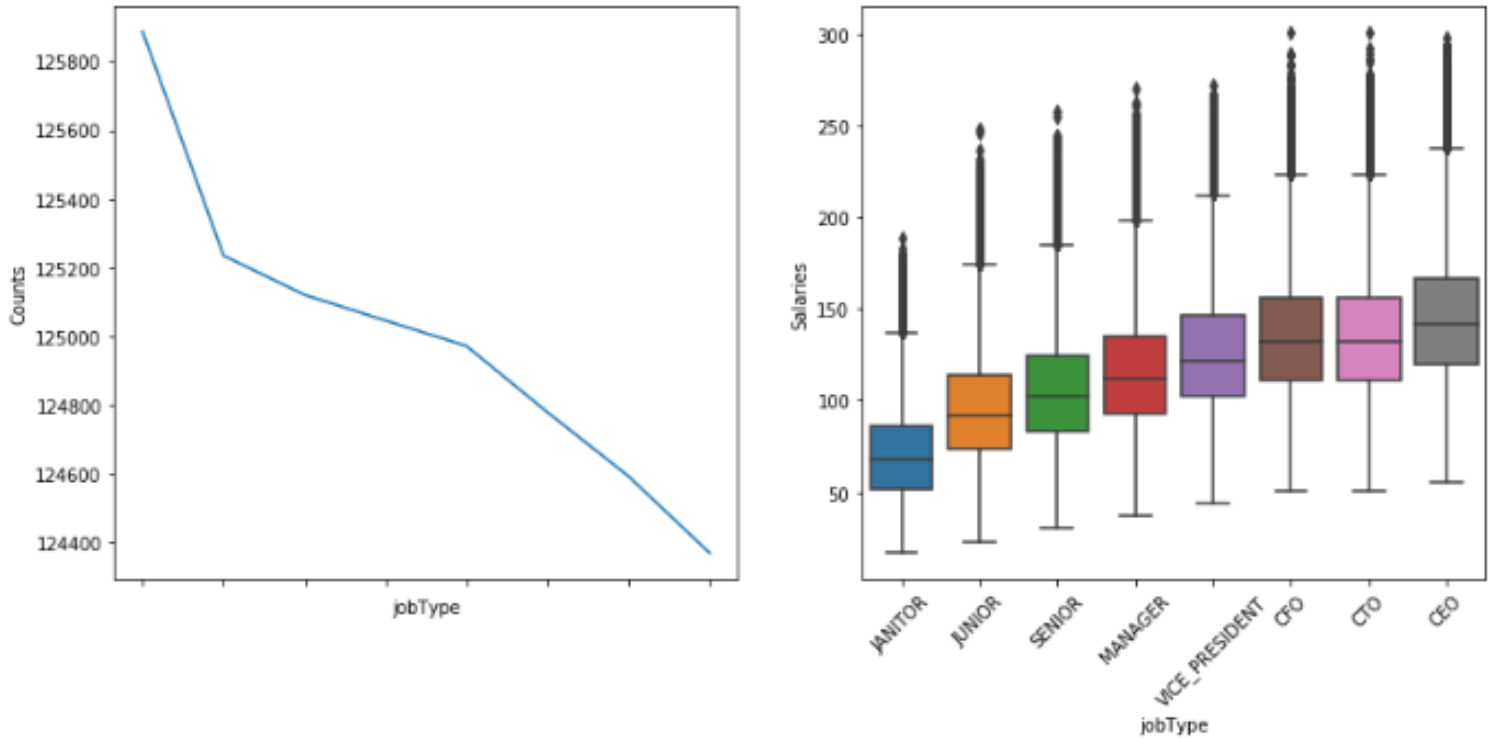
Discover the data (through EDA)



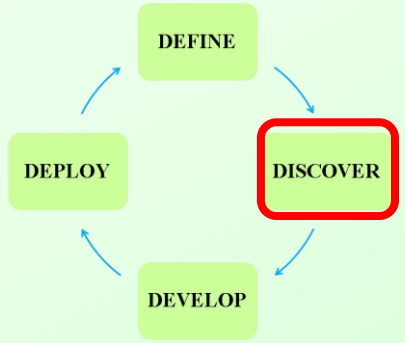
- **Step 9: Plot all the features separately (‘jobType’)**

- This shows that the more higher the role, the more salary the person is earning, which is totally reasonable and believable.
- This feature has definitely impact on the target (salary) and hence will be kept as one of the features in the training dataset.

```
plot_feature(train_df, 'jobType')
```



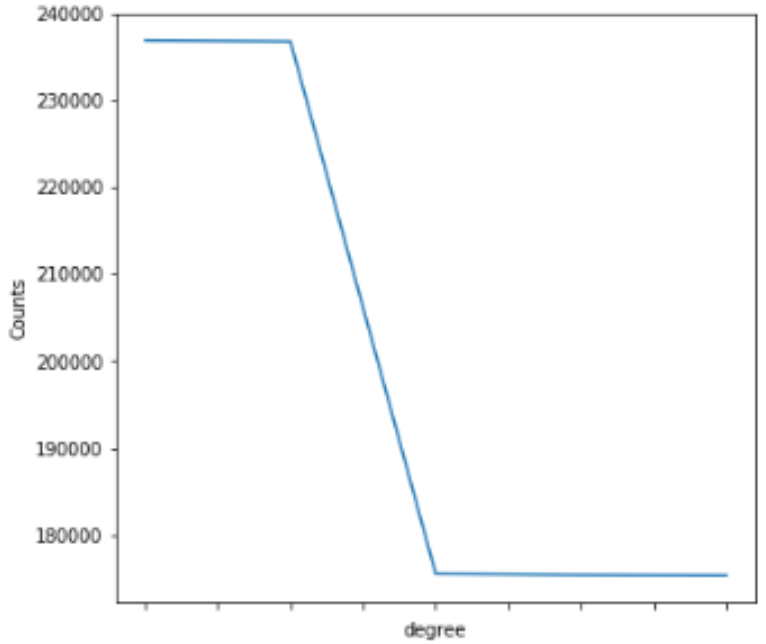
Discover the data (through EDA)

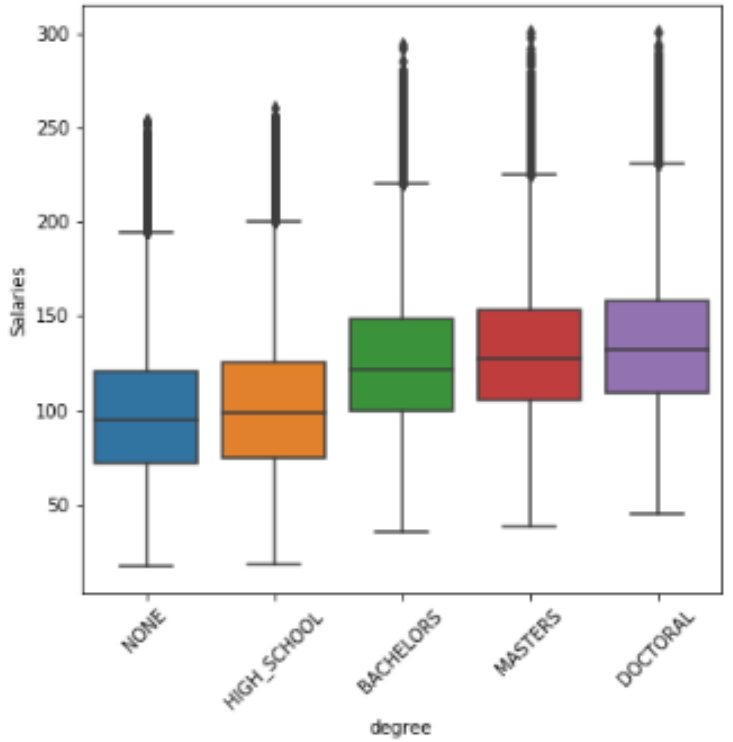


- **Step 9: Plot all the features separately (‘degree’)**

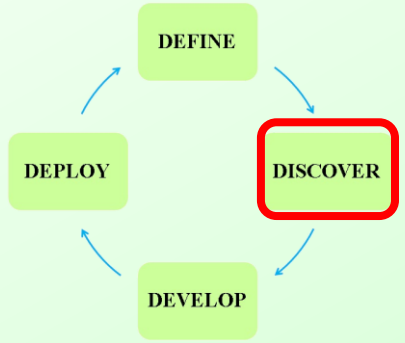
- This shows that the more the degree level, the more salary the person is earning, which is completely acceptable and makes sense.
- This feature has definitely impact on the target (salary) and hence will be kept as one of the features in the training dataset.

```
plot_feature(train_df, 'degree')
```



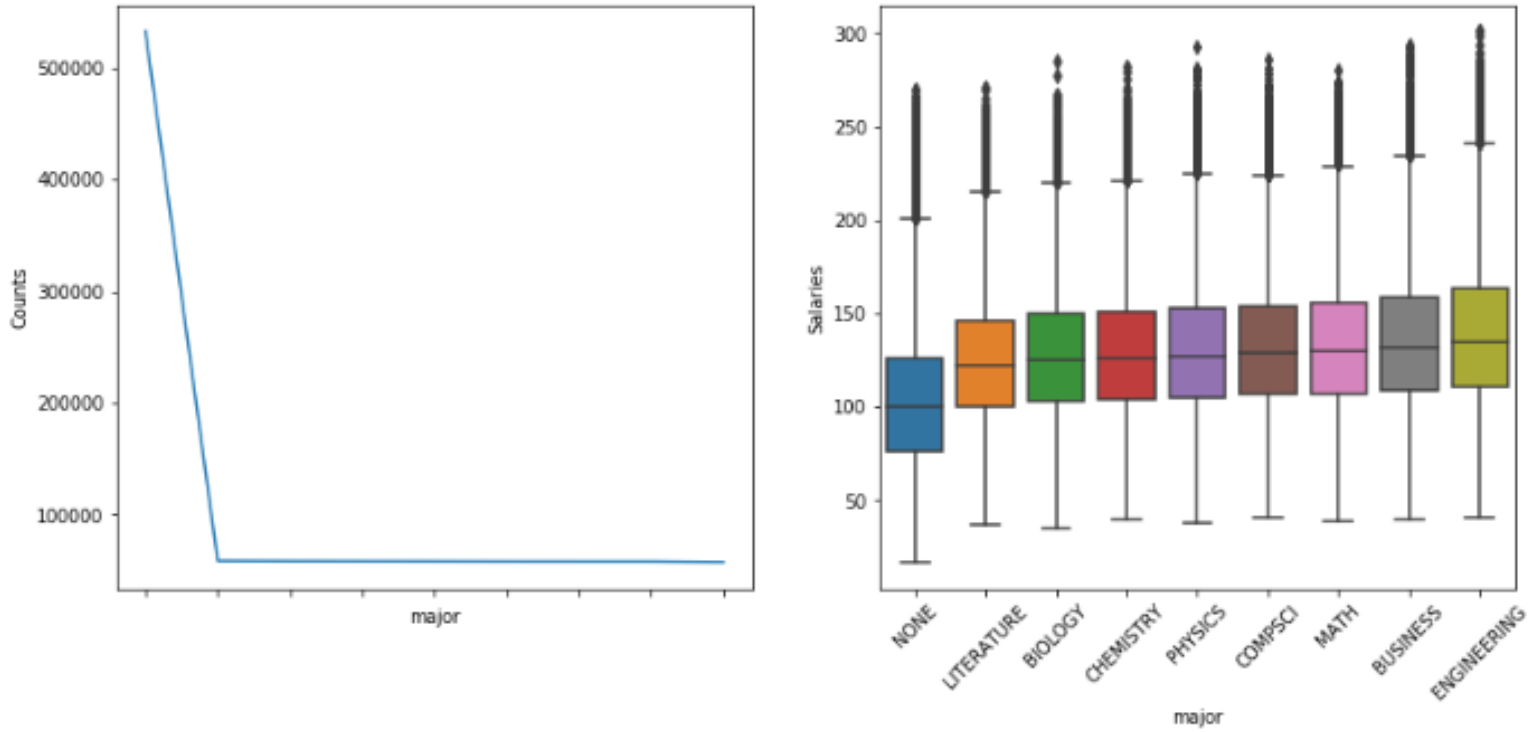


Discover the data (through EDA)



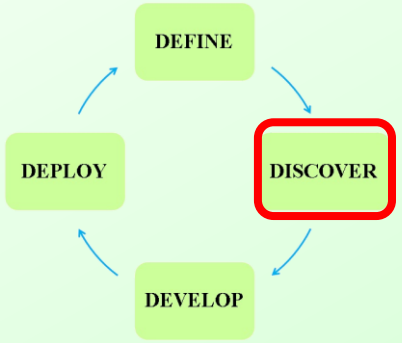
- **Step 9: Plot all the features separately ('major')**

```
plot_feature(train_df, 'major')
```



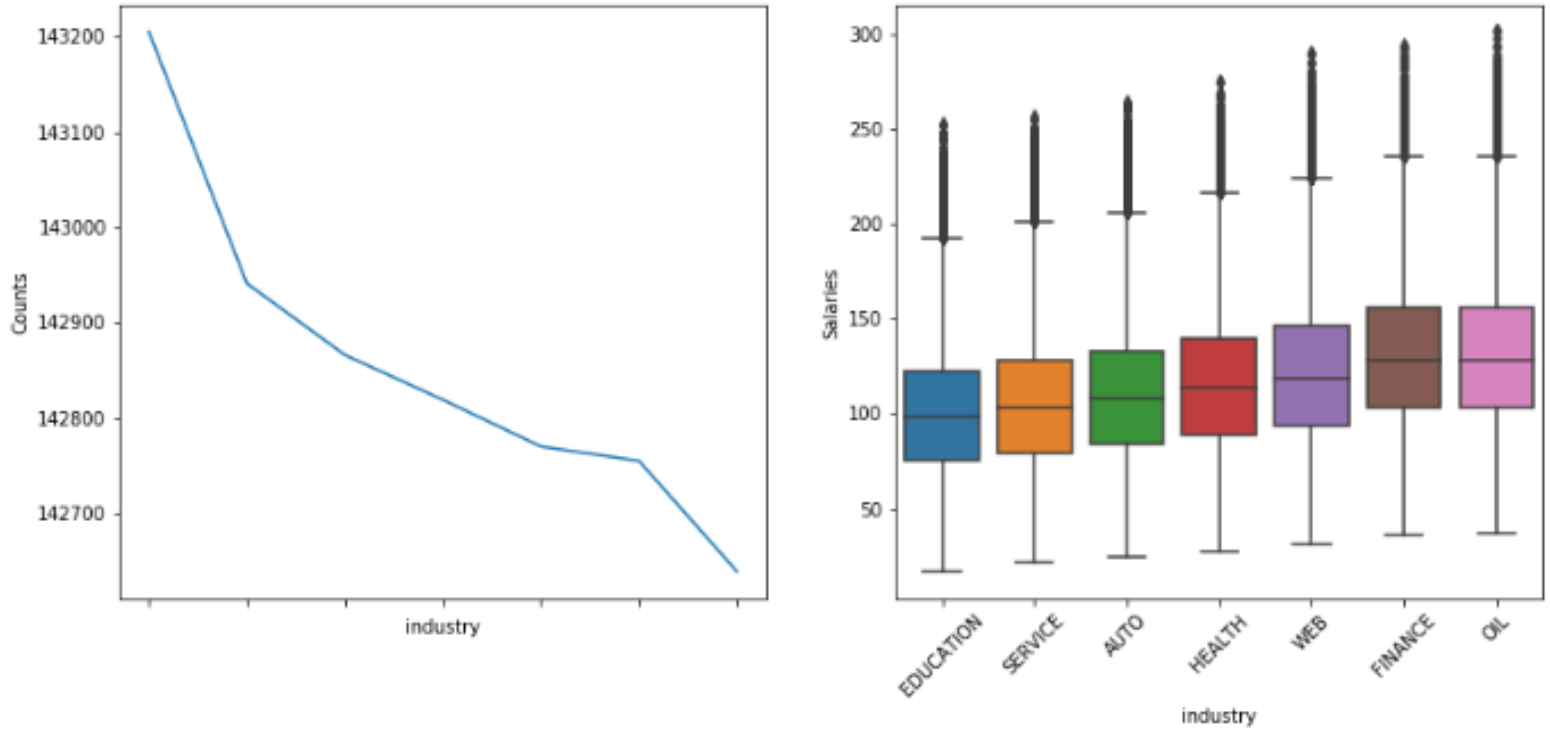
- Apart from people with NONE degree, the rest of the degrees are earning pretty much the same salary. However, the salaries of engineering, business and maths are slightly higher than other.
- This feature has impact to some extent on the target (salary) and hence will be kept as one of the features in the training dataset.

Discover the data (through EDA)



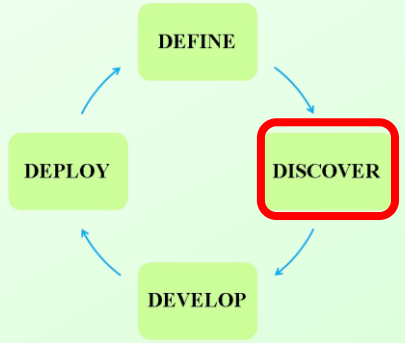
- **Step 9: Plot all the features separately (‘industry’)**

```
plot_feature(train_df, 'industry')
```



- This shows, as expected, that the oil, finance and web has highest salaries. Also, this shows that different industries have different salaries.
- These 2 observations are reasonable and rational.
- This feature has definitely impact on the target (salary) and hence will be kept as one of the features in the training dataset.

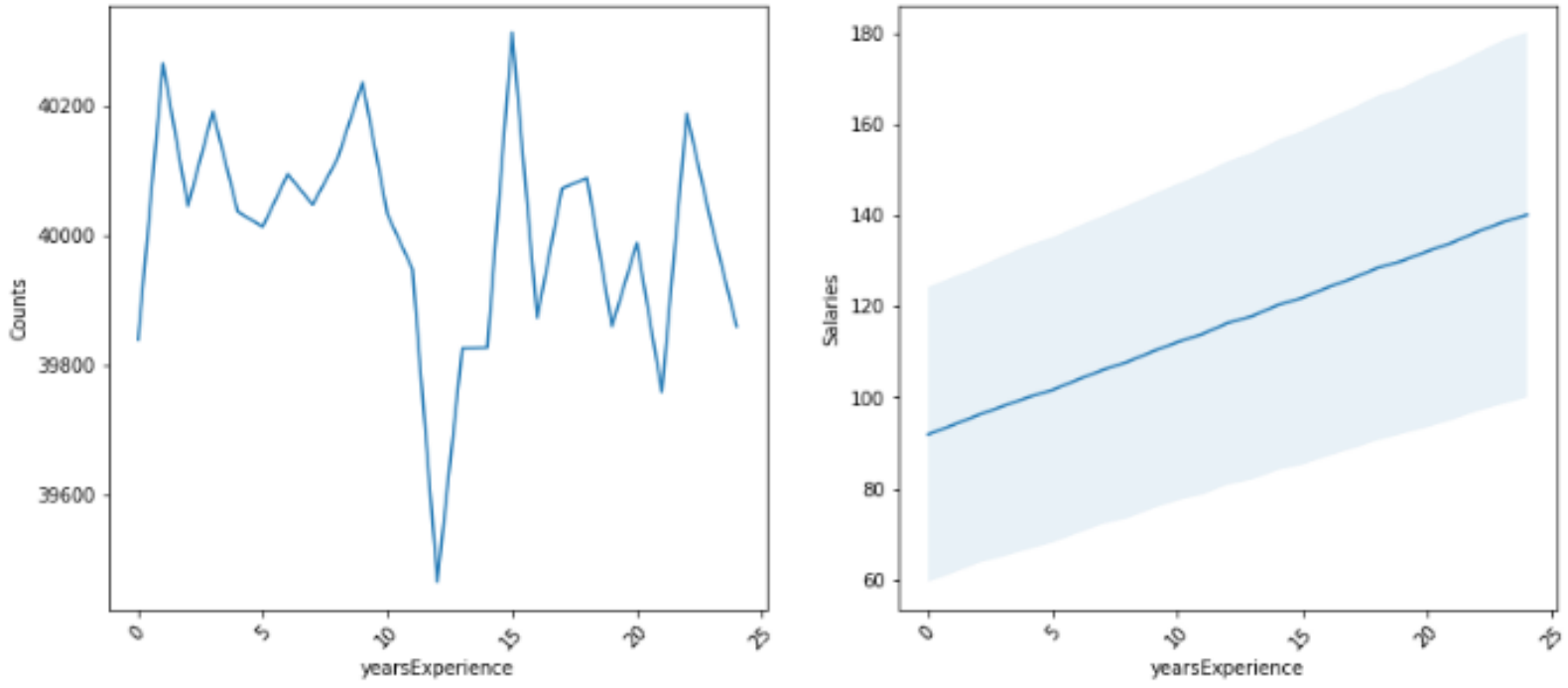
Discover the data (through EDA)



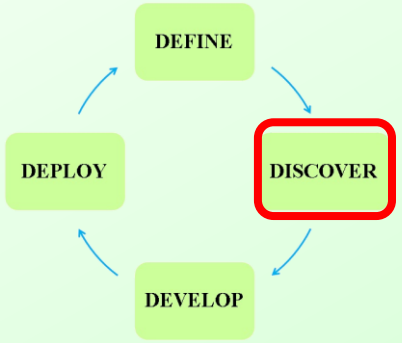
- **Step 9: Plot all the features separately (‘yearsExperience’)**

- This shows, as expected, that the more experience the person has, the more salary they earn, which is completely genuine and logical.
- This feature has definitely impact on the target (salary) and hence will be kept as one of the features in the training dataset.

```
plot_feature(train_df, 'yearsExperience')
```



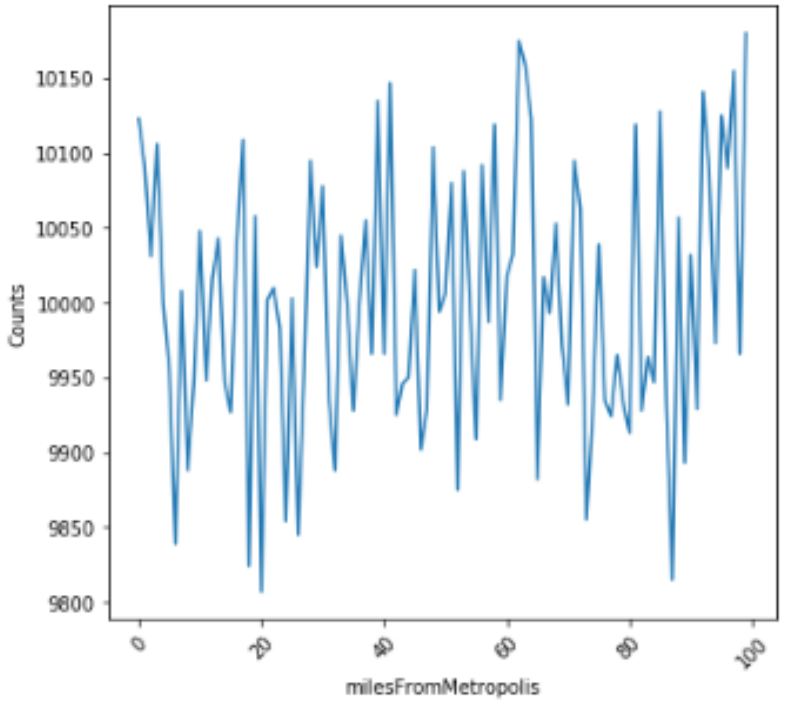
Discover the data (through EDA)

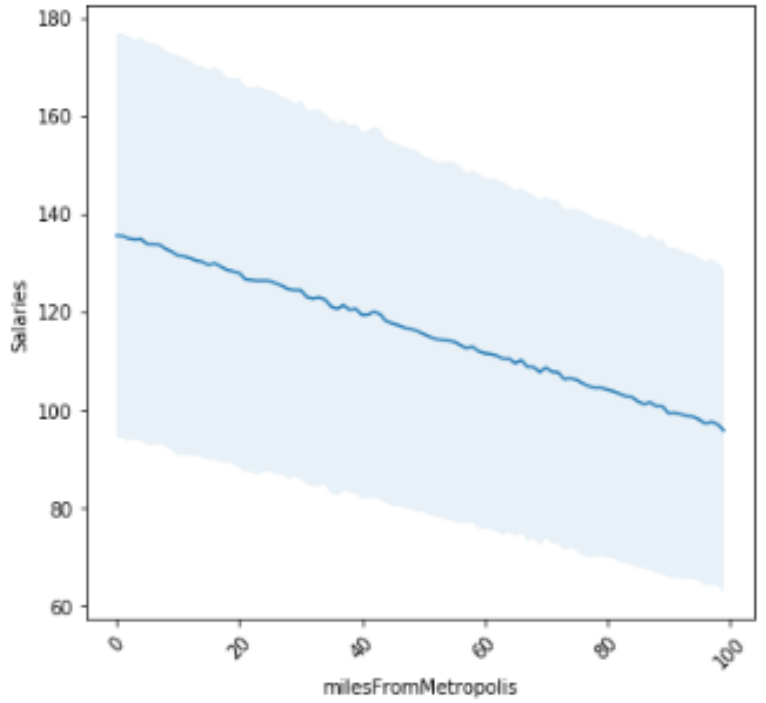


- **Step 9: Plot all the features separately (‘milesFromMetropolis’)**

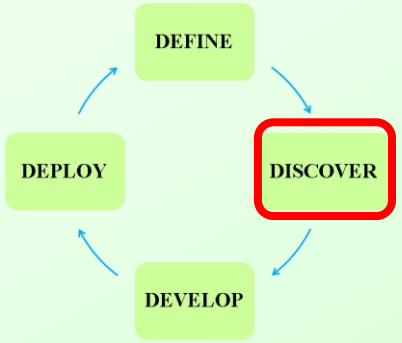
- This shows an inverse relationship between the miles from Metropolis and the salary, which makes perfect sense.
- This feature has definitely impact on the target (salary) and hence will be kept as one of the features in the training dataset.

```
plot_feature(train_df, 'milesFromMetropolis')
```





Discover the data (through EDA)

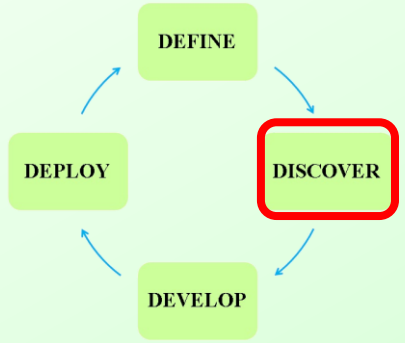


- **Step 10: Identification of correlation between all the features and target**
- **Step 10.1: Label encoding the categorical variables**
  - **Purpose:**
    - **To be able to plot numeric features against categorical features (for Step 10.2)**

```
def encode_label(df, col):
    #encode the categories using average salary for each category to replace label
    cat_dict ={}
    cats = df[col].cat.categories.tolist()
    for cat in cats:
        cat_dict[cat] = train_df[train_df[col] == cat]['salary'].mean()
    df[col] = df[col].map(cat_dict)
```

```
for col in train_df.columns:
    if train_df[col].dtype.name == "category":
        encode_label(train_df, col)
```

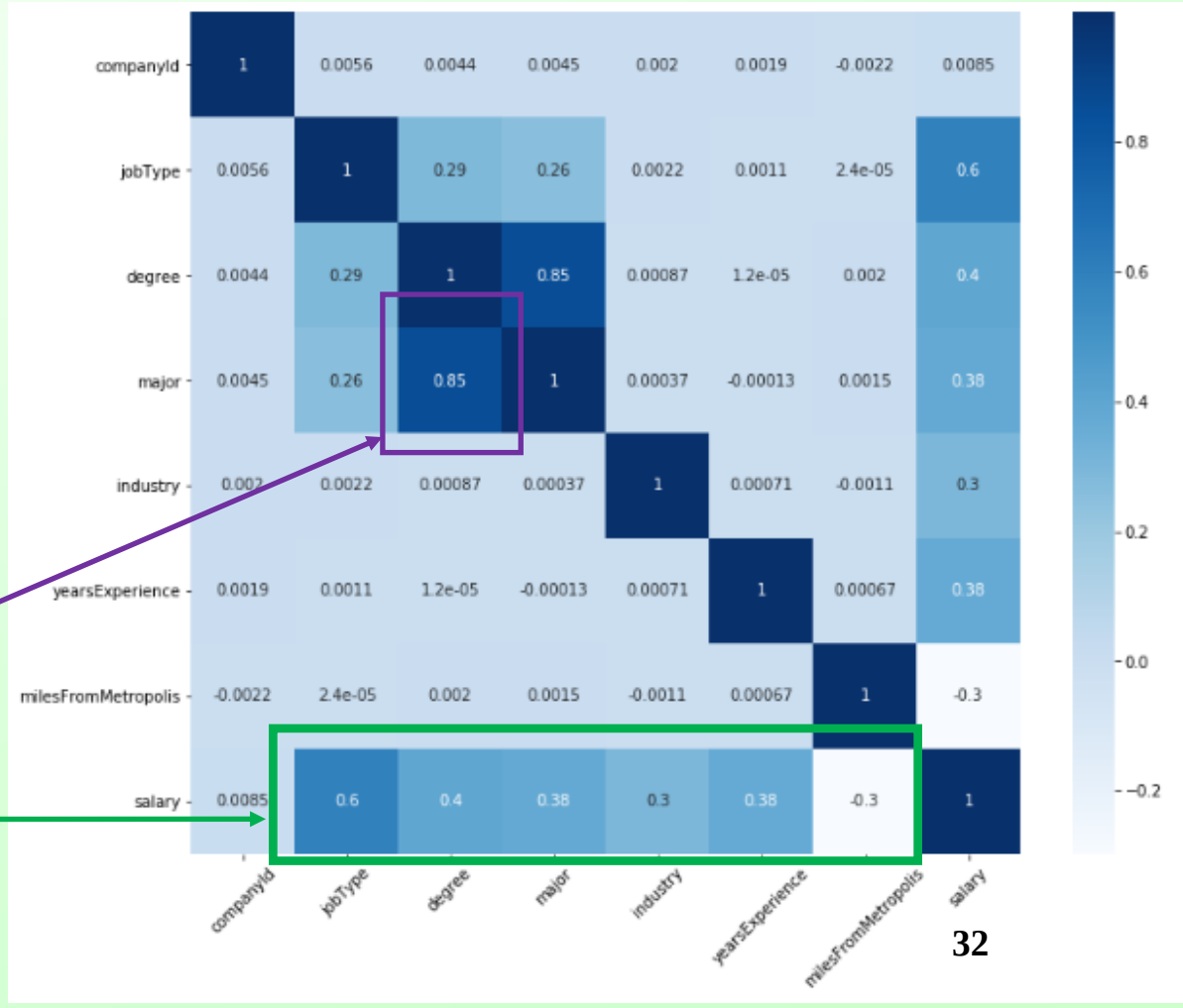
Discover the data (through EDA)



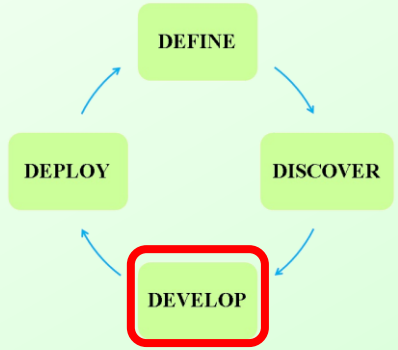
- **Step 10.2: Develop a map of all the features and target**
  - **Purpose:**
    - **To inspect if there is any correlation between the features**
    - **To select appropriate type of models to fit to the data based on any potential correlation**

Some degree of correlation between major and degree.

This shows that the salaries are correlated to all the features except companyId.

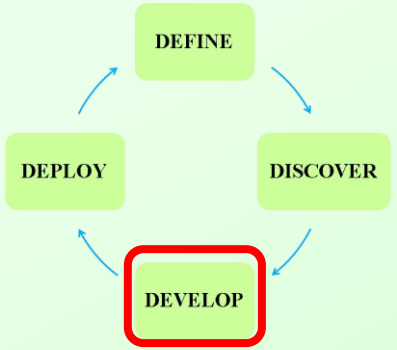


Develop models



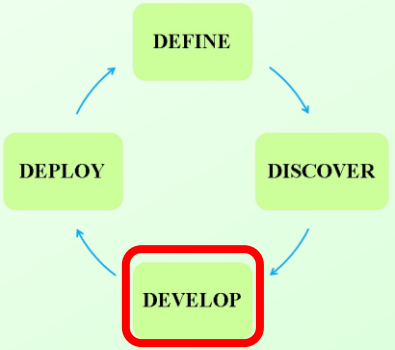
- **Feature engineering**
  - Feature engineering is implemented to inform the model of some of the similarities in the data and hence to boost the performance of the models.
  - All the salaries that had the same ‘companyId’, ‘jobType’, ‘degree’, ‘major’, and ‘industry’ were grouped.
  - For each aforementioned group, the mean, minimum, maximum, standard deviation and median was calculated as a new dataset with columns as ‘group_mean’, ‘group_min’, ‘group_max’, ‘group_std’ and ‘group_median’ was created.
  - The aforementioned dataset was merged with the original dataset and was considered as the training dataset.
  - It was carefully noticed that the features that are engineered on the training set are also applicable to the test dataset.

Develop models



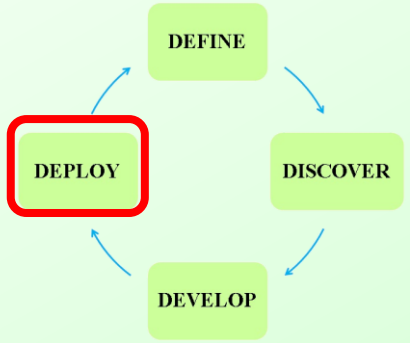
- **Selection of the models**
  - Three regression models were selected to predict the target variable (salary), based on all the features (including the features engineered):
    - Linear Regression
    - Random Forest Regressor
    - Gradient Boosting Regressor
  - The Mean Squared Error (MSE) was used as the metric to evaluate and compare the performance of these three models.

Develop models

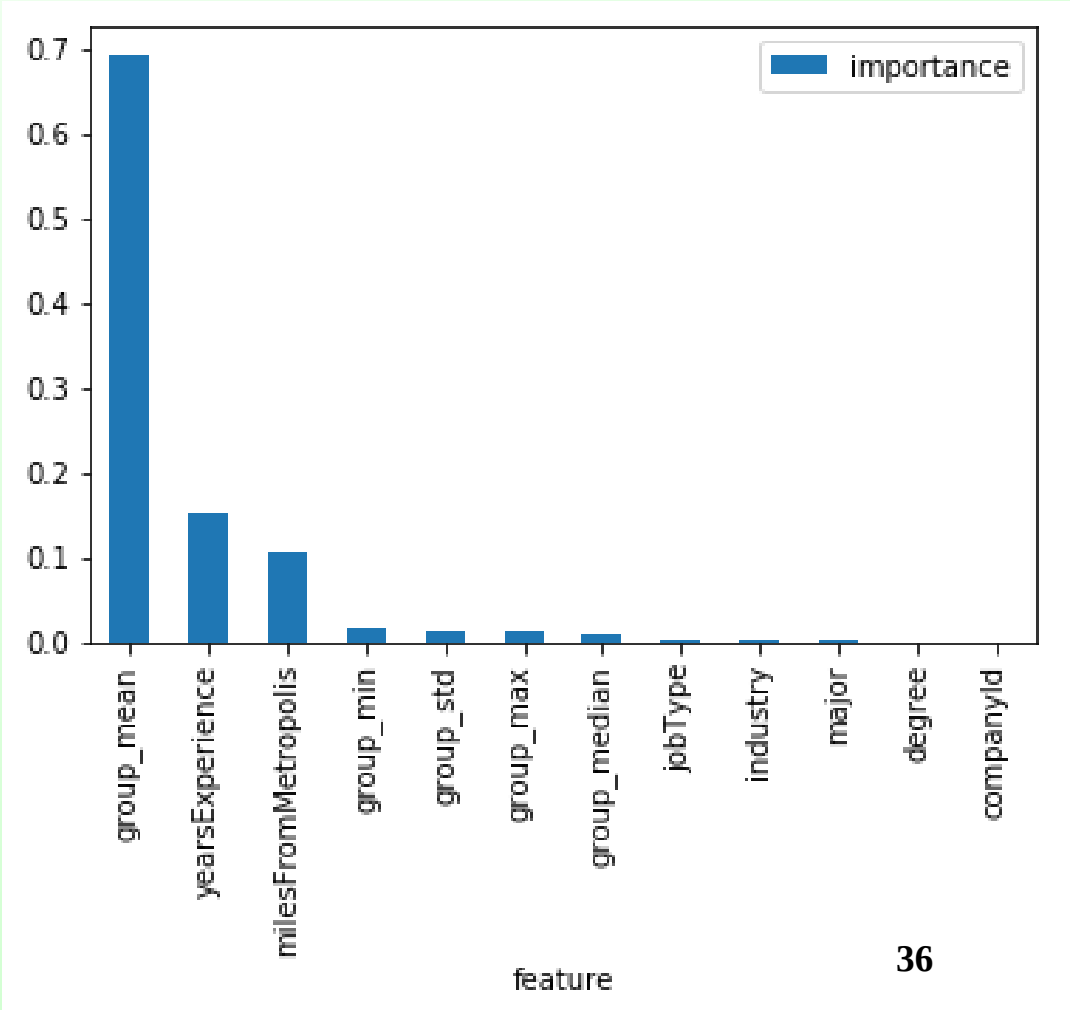


- **Performance of the models**
  - The value of Mean Squared Error (MSE) for the selected models are:
    - Linear Regression: 358.17
    - Random Forest Regressor: 313.63
    - Gradient Boosting Regressor: 313.10

Deploy the best model



- **Selection of the Best model:**
  - The Gradient Boosting Regressor with the value of MSE of 313.10 is considered as the best model.
  - As can be seen, the following features are the key predictors (have the most importance/impact on the value of the salary):
    - group_mean
    - yearsExperience
    - milesFromMetropolis



Next steps

- Feature engineering can be extended to also consider ‘yearsExperience’ and ‘milesFromMetropolis’.
- The performance of the models could have been improved having further features in the original dataset, i.e. ‘recruitType’ (‘contract’ or ‘permanent’), ‘contractType’ (‘full-time’, ‘part-time’), etc.

Thank You!

Seyed Hamid Reza Hosseini
Newcastle upon Tyne, UK
June 2020
hosseini.shr@gmail.com