CAPSTONE PROJECT

HR ANALYTICS

USING DATA TO INFORM // TRANSFORM // AND EMPOWER HR DECISIONS

PGP-DSE Gurgaon Jan'21-Group 7
Siddhanth Rawat
Pragyansh Sharma
Ritu Dhounchak
Divyanshi Verma
Urvashi Rawat

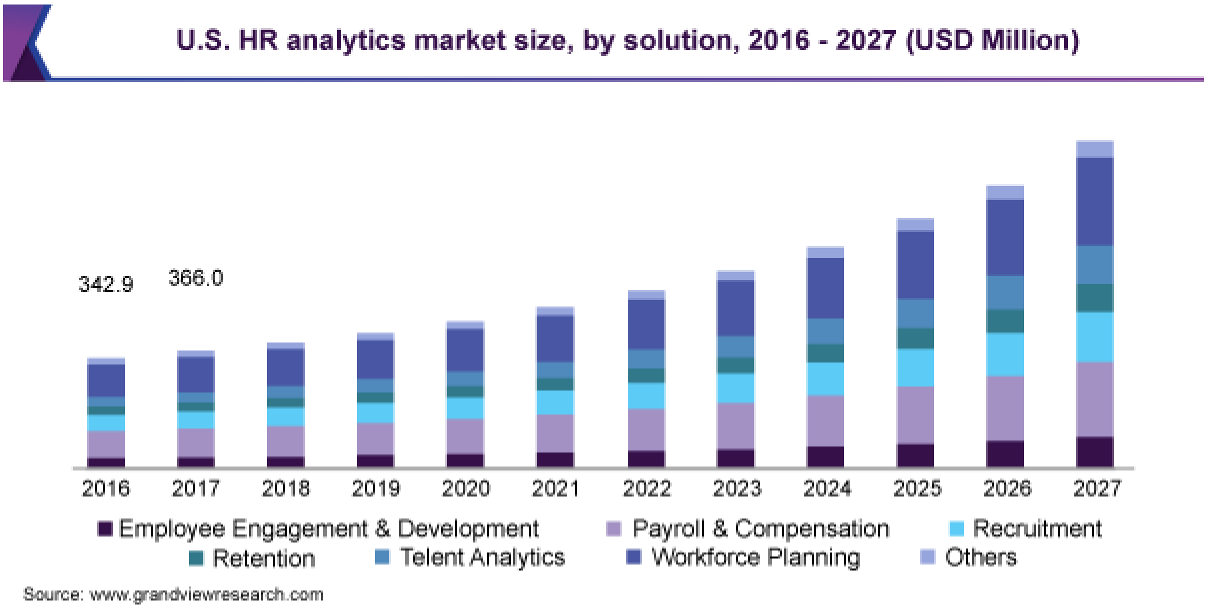
Mentored By: Mr. Vikash Chandra

Table of Contents:

Page No.

Industry Review:

The global HR analytics market size was valued at USD 2.25 billion in 2019 and is expected to grow at a compound annual growth rate (CAGR) of 14.2% from 2020 to 2027. The growing need for improving the efficiency and reduce the operational cost in an organization is likely to drive the demand for HR analytics across various industries. It allows the organization to promote workforce optimization and create metrics for full workforce performance improvement. It can analyze and access huge unstructured data in real-time to implement business decisions related to human resource utilization, which is crucial for business development.



U.S. HR analytics market size, by solution, 2016 - 2027 (USD Million)

Source: www.grandviewresearch.com

sample report includes market data points, ranging from trend analyses to market estimates & forecasts. Growing need to employ skilled talent and enhance employee retention with the help of innovative technologies such as Internet of Things (IoT) and analytics is projected to push the market growth. With the help of these solutions, the companies can focus more on improving employee engagement, easy employee on-boarding, and increasing workforce efficiency. These analytical solutions allow HR professionals to manage, attract, and retain employees, which can subsequently lead to an increased Return on Investment (ROI) for the organization, help in improving productivity, and provide a better work environment.

Human resource management solution providers are focusing more on current technological trends in the fields of Internet of Things (IoT) and Artificial Intelligence (AI) to improve their software designs as per the customer requirements. Many companies are investing in R&D activities to provide solutions that are easy to understand, deploy, and offer increased efficiency for the clients. For instance, major solutions providers, such as Oracle; SAP SE; and Workday Inc. offer their products on cloud in order to reduce the need for upgrading systems frequently. Furthermore, Government initiatives to support the adoption of digital technologies, mobile applications, and internet connectivity to reduce manual efforts in a bid to enhance the productivity of organizations, are expected to fuel the market growth.

Over the last decade, there is a vast amount of workforce data available with the organizations, which, if used well, can lead to significant conclusions pertaining to an organization's decisions. Rapid technological advancements in the fields of analytics and availability of workforce data have resulted in many organizations applying these technologies in their business processes. Recruiting and training employees is an expensive process and if the employee leaves the organization without serving for a significant period it becomes a loss for the organization. Hence, employee retention, talent management, and succession planning, among others have become critical factors for improving the overall organizational performance. The use of HR analytics helps organizations in better decision-making and maximizing their capital, which is subsequently accelerating the growth of the HR analytics market.

Although the HR analytics offers several benefits such as cost-effectiveness and ease of deployment, among others, security concerns pose a threat to the market growth. Small and Medium Enterprises (SMEs) are increasingly adopting public cloud services due to its low cost, thus increasing the possibility for data breaches, cyber-attacks, and so on. Hence, the security issues related to using cloud service is the major challenging factor hindering the adoption of HR analytics. However, if efforts are made between HR management teams and IT teams to agree upon a set of technical controls then this could result in the prevention of cyber-attacks and data loss.

Challenges for HR Analytics

1. Holding Information: HRIS, LMS, ATS, etc. are chock full of ripe, relevant, and often current information. Yet, it is not common practice that HR delivers high-impact reports to stakeholders.
2. Broad Data: Focusing on metrics too big to assess. E.g. Voluntary vs. Involuntary Termination statistics. What if the right people are voluntarily leaving? What if the wrong employees are staying? Another example: EEO statistics are provided to the Dept. of Labor annually (for most organizations), but if an organization is charged with discrimination, data relating to specific managers are also required. Is this information reviewed?
3. Analysis Skills: HR is often not the analytical expert in the organization; instead it was the CFO or Engineering or Quality team. Enhancing analysis and data evaluation skills may be a professional development focus for some HR professionals.
4. Big Data and NOT So Big: If the organization is large, it is easy to collect big data on recruiting, training, succession planning, turnover. But organizations that are smaller do not often have big data. Therefore analysis and critical thinking related to smaller data require different approaches when small samples are used to make big decisions.
5. Assessments in Hiring: Uniform Guidelines on Employee Selection Procedure (from Dept. of Labor) requires all processes/practices used to hire/promote to be valid. Too often, assessments or practices are used that have not been internally validated (tests, assessments, interview practices, review of applications).
6. Curiosity: Ask, “What is the most important thing(s) needed to be evaluated to impact the business and to help leaders make informed decisions?”

1. DATA PRE PROCESSING:

Our Dataset has rows and 14 Columns,

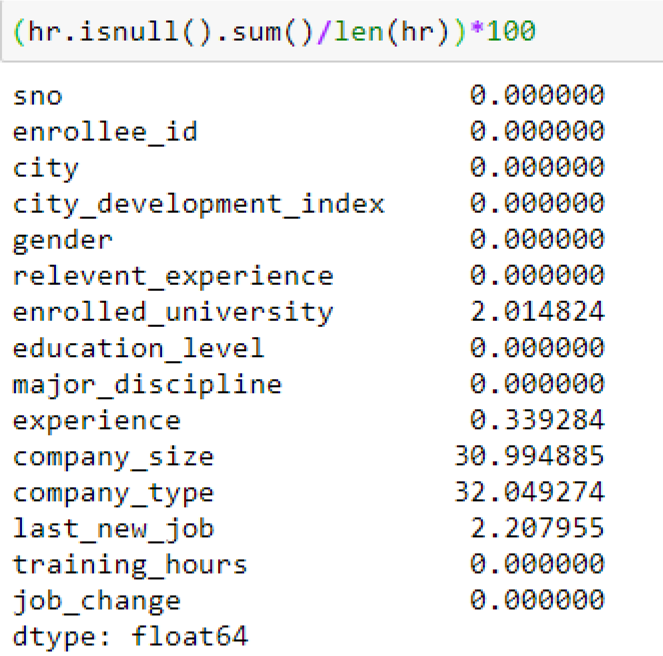
This dataset, we don't have any null values hence our dataset is free from null values.

S.No.	Column Name	Column Description	D-Type
1	Enrollee_id	Unique ID for candidate	int64
2	City	City code	int64
3	City_ development_index	Development index of the city	object
4	Gender	Gender of candidate	float64
5	Relevent_experience	Relevant experience of candidate	object
6	Enrolled_university	Type of University course enrolled if any	object
7	Education_level	Education level of candidate	object
8	Major_discipline	Education major discipline of candidate	object
9	Experience	Candidate total experience in years	float64
10	Company_size	No of employees in current employer's company	object
11	Company_type	Type of current employer	object
12	Last_new_job	Difference in years between previous job and current job	object
13	Training_hours	training hours completed	int64
14	Job Change	Look ing for a job change or not	object

1. DATA CLEANING:

Data cleaning is the process of spotting and correcting inaccurate data and it prepares the data for analysis by removing or modifying the data that is incorrect, incomplete, irrelevant, duplicated, or missing. So, while cleaning the data we have dropped the irrelevant columns from the dataset and treated the missing values.

1.1 NULL VAlUE IMPUTATION:



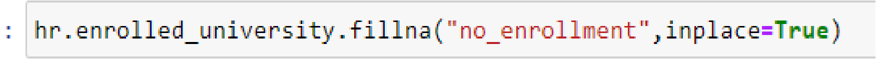
```
(hr.isnull().sum()/len(hr))*100
```

```
sno                        0.000000
enrollee_id                0.000000
city                       0.000000
city_development_index     0.000000
gender                     0.000000
relevent_experience        0.000000
enrolled_university        2.014824
education_level            0.000000
major_discipline           0.000000
experience                 0.339284
company_size              30.994885
company_type              32.049274
last_new_job               2.207955
training_hours             0.000000
job_change                 0.000000
dtype: float64
```

- The columns that contain maximum number of null values are in the columns company_type wghich 32% , followed by company_size which is 30% .
- The columns enrolled_university, education_level, experience has null values less than 5%.

1. NULL VALUE IMPUTATION FOR ENROLLED UNIVERSITY

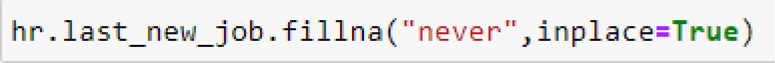
The percentage of null values in Enrolled Univeristy column was only 2 % hence mode imputation technique was used to impute missing values.



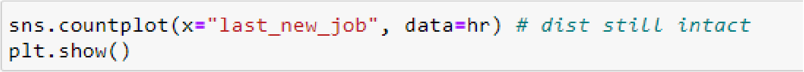
```
: hr.enrolled_university.fillna("no_enrollment",inplace=True)
```

2. NULL VALUE IMPUTATION FOR LAST_NEW_JOB

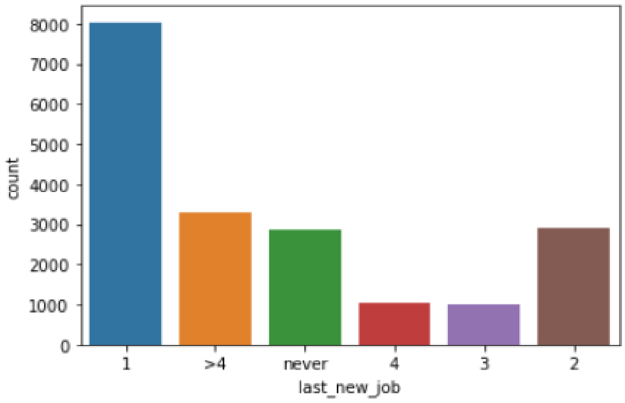
The percentage of null values in last_new_job columns was also less than 5%. However, the columns already had value never for the employees with no year gap between their last employement and current employement. Therefore, missing values were imputed using “never” which also kept the overall distribution of the data intact.



```
hr.last_new_job.fillna("never",inplace=True)
```



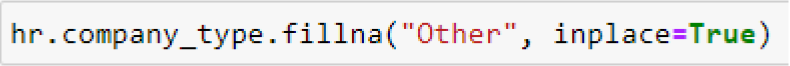
```
sns.countplot(x="last_new_job", data=hr) # dist still intact
plt.show()
```



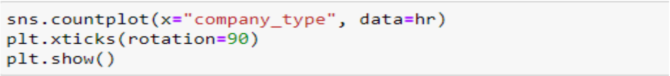
3. NULL VALUE IMPUTATION FOR COMPANY_TYPE

The percentage of null values in Company_type columns was 32% and the columns already had value “Other” for the employees which had different company type.

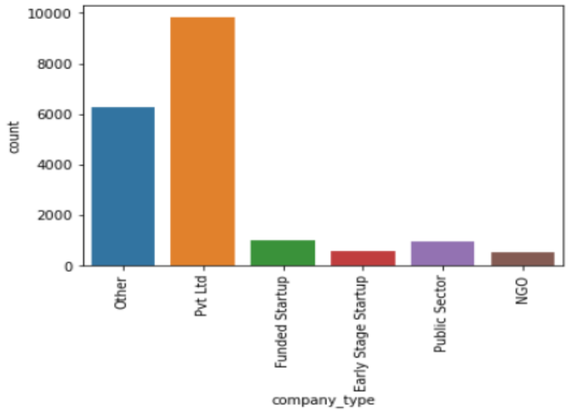
Therefore, missing values were imputed using “Other” which also kept the overall distribution of the data intact.



```
hr.company_type.fillna("Other", inplace=True)
```

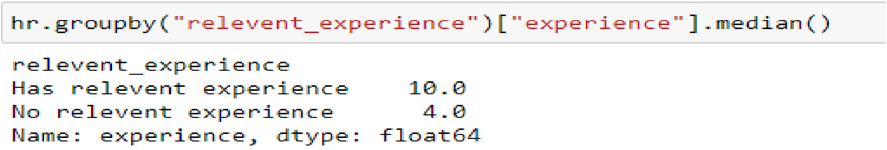


```
sns.countplot(x="company_type", data=hr)
plt.xticks(rotation=90)
plt.show()
```



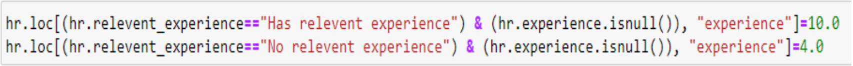
4. NULL VALUE FOR EXPERIENCE;

In order to impute the null values in experience columns the experience column was divided into two categories “relevant experience” and “non-relevant experience” using columns “RELEVANT EXPERIENCE”. After categorization, the median for both the categories was computed and used to impute missing values.



```
hr.groupby("relevent_experience")["experience"].median()
```

```
relevent_experience
Has relevent experience     10.0
No relevent experience       4.0
Name: experience, dtype: float64
```

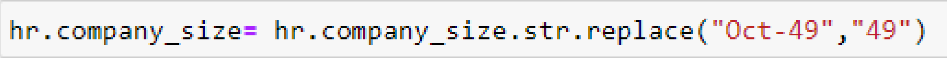


```
hr.loc[(hr.relevent_experience=="Has relevent experience") & (hr.experience.isnull()), "experience"]=10.0
hr.loc[(hr.relevent_experience=="No relevent experience") & (hr.experience.isnull()), "experience"]=4.0
```

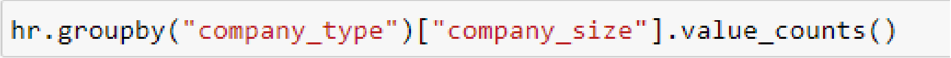
5. NULL VALUE FOR COMAPNY SIZE

In order to impute the null values in "company size” column, firstly the value of “Oct-49” was changed to 49 to make the computation process easy, then the column was divided into six categories using values of column "Company_type".

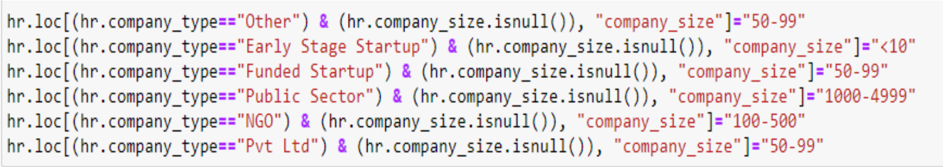
After categorization, the most frequent value for each category were computed and used to impute nulls.



```
hr.company_size= hr.company_size.str.replace("Oct-49","49")
```



```
hr.groupby("company_type")["company_size"].value_counts()
```



```
hr.loc[(hr.company_type=="Other") & (hr.company_size.isnull()), "company_size"]="50-99"
hr.loc[(hr.company_type=="Early Stage Startup") & (hr.company_size.isnull()), "company_size"]="<10"
hr.loc[(hr.company_type=="Funded Startup") & (hr.company_size.isnull()), "company_size"]="50-99"
hr.loc[(hr.company_type=="Public Sector") & (hr.company_size.isnull()), "company_size"]="1000-4999"
hr.loc[(hr.company_type=="NGO") & (hr.company_size.isnull()), "company_size"]="100-500"
hr.loc[(hr.company_type=="Pvt Ltd") & (hr.company_size.isnull()), "company_size"]="50-99"
```

1.2 SELECTING RELEVANT FEATURES

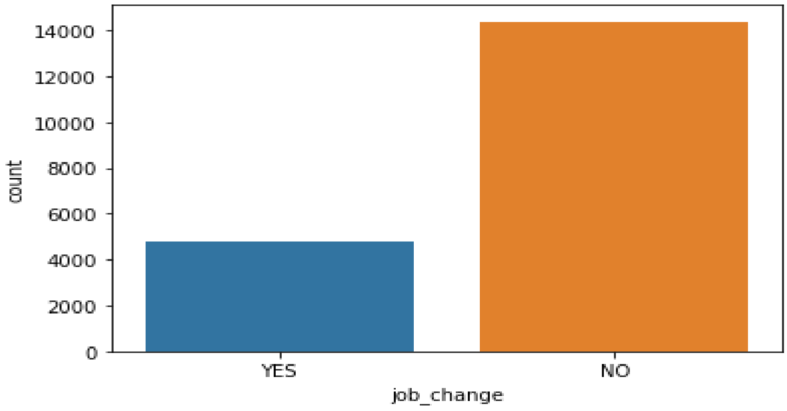
S.No.	Column Name	D-Type
1.	City_ development_index	object
2.	Gender	float64
3	Relevent_experience	object
4	Enrolled_university	object
5	Education_level	object
6	Major_discipline	object
7	Experience	float64
8	Company_size	object
9	Company_type	object
10	Last_new_job	object
11	Training_hours	int64
12	Job Change	object

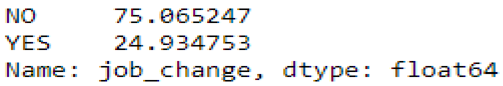
2. EXPLORARTORY DATA ANLAYSIS

2.1 Univariate Analysis

Categorical Variables:

Job Change: The percentage ratio for job change of No to Yes is 24:75.



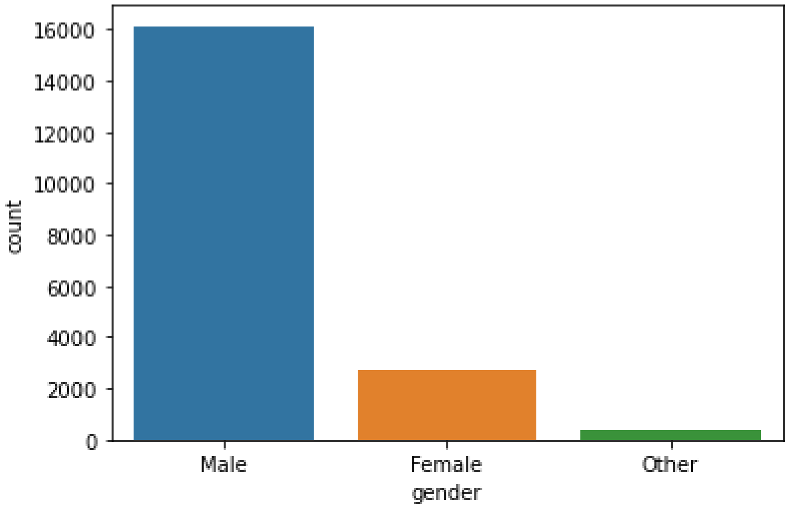


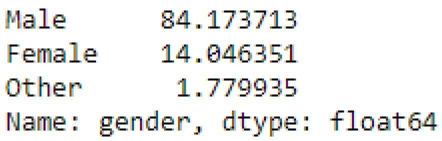
```
NO      75.065247
YES     24.934753
Name: job_change, dtype: float64
```

Gender:

The no. of employees working in diff sectors are mostly males than females followed by others.

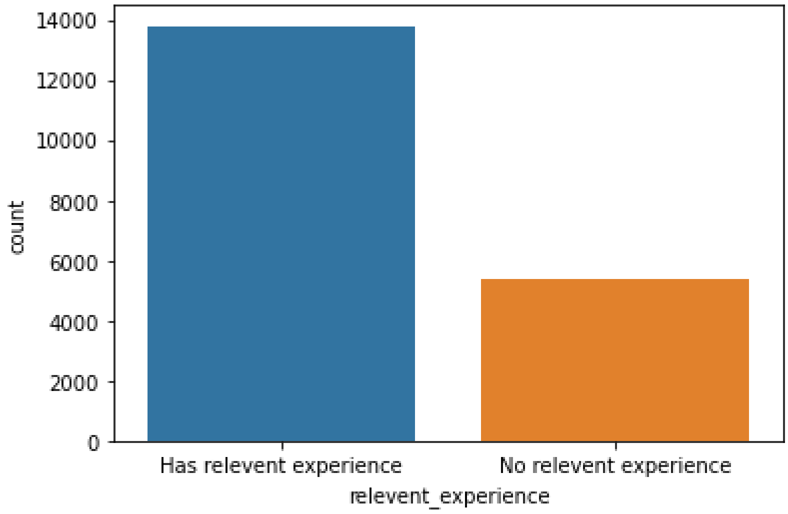
The gender looking for job change in increasing order is given by: other> female> male





```
Male      84.173713
Female    14.046351
Other      1.779935
Name: gender, dtype: float64
```

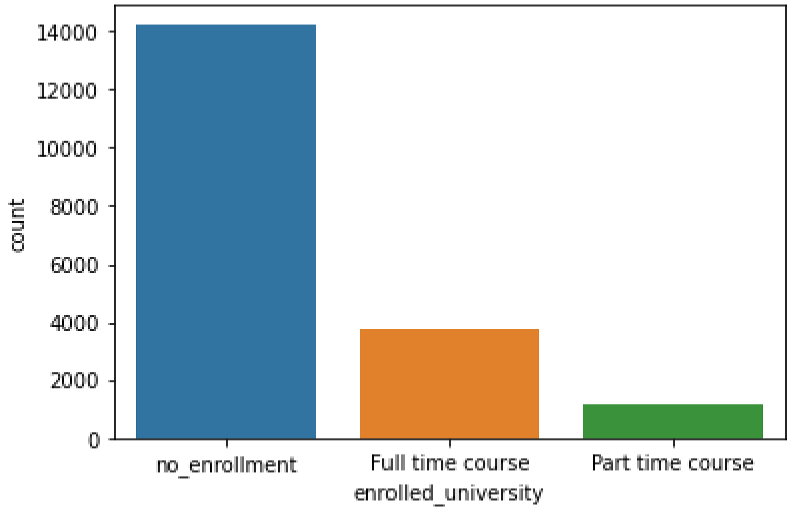
Relevant Experience: People with no relevant experience are more likely to look for jobs.



```
Has relevent experience    71.990813
No relevent experience     28.009187
Name: relevent_experience, dtype: float64
```

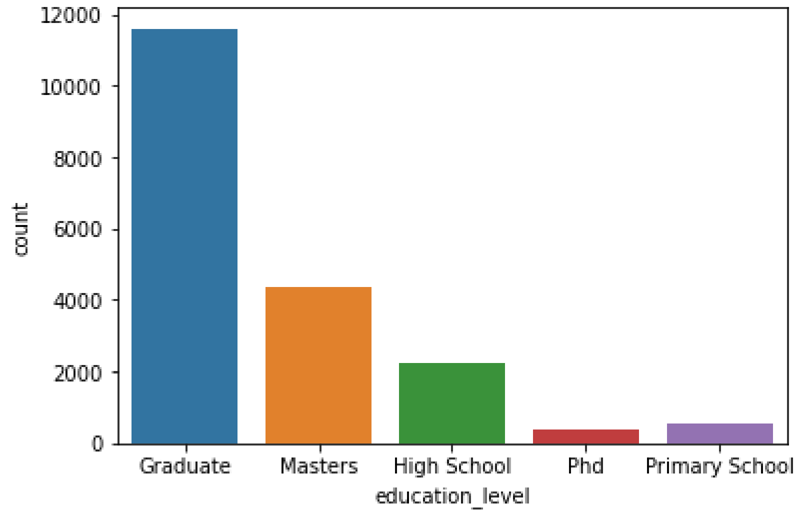
Enrolled University:

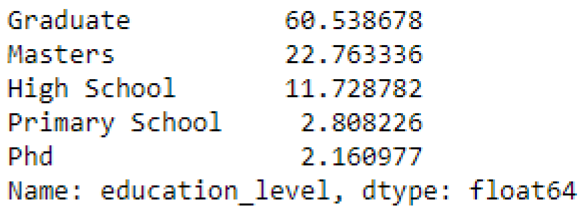
Maximum employees are not enrolled in any courses and some of the employees are enrolled in Full-time and Part-time courses.



```
no_enrollment       74.136131
Full time course    19.610607
Part time course     6.253262
Name: enrolled_university, dtype: float64
```

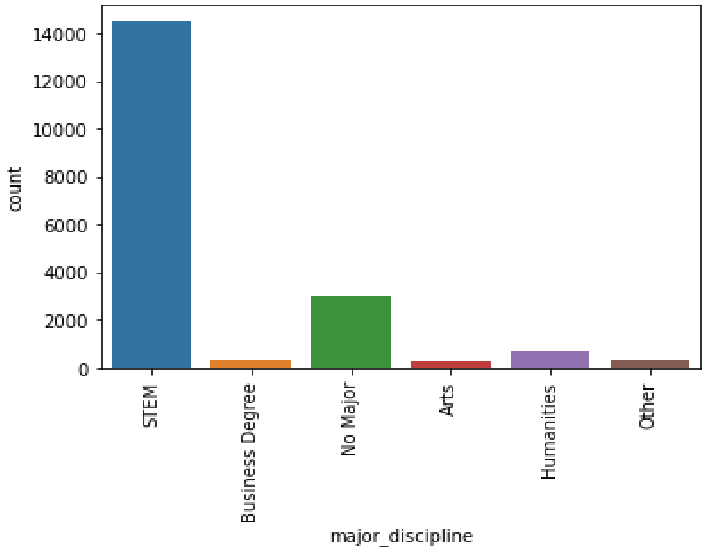
Education Level: The max no. of candidates have graduate degrees followed by masters and the lowest level of education is primary school.

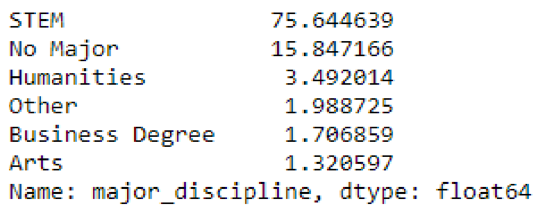




```
Graduate          60.538678
Masters           22.763336
High School       11.728782
Primary School     2.808226
Phd                2.160977
Name: education_level, dtype: float64
```

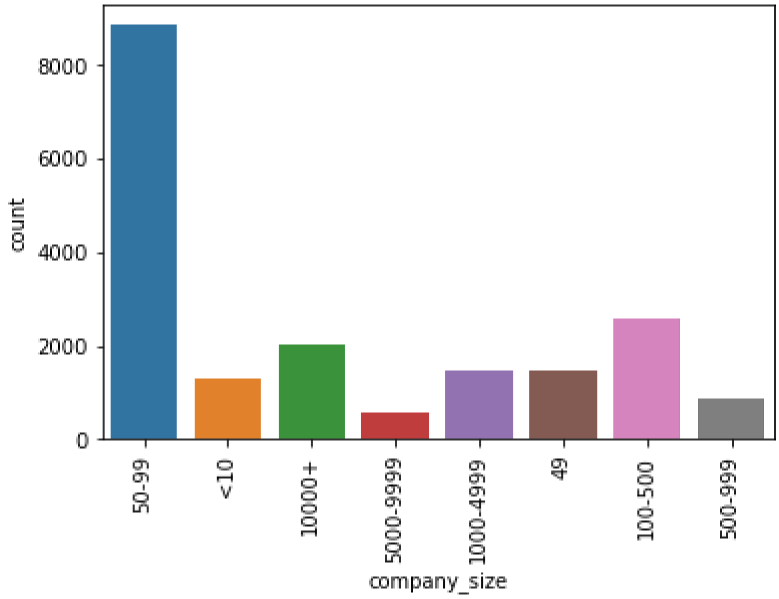
Major Discipline: The 30% graduates with other as a major discipline are actively looking for job change followed by 28% of graduates with the stem as major discipline as compare to another educational level.

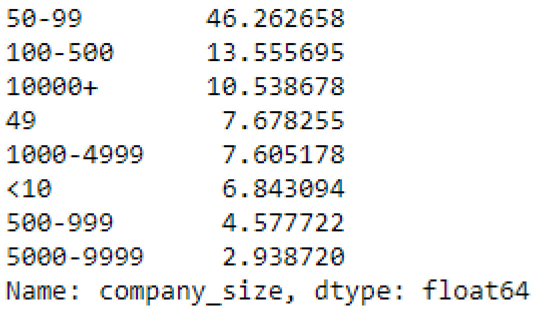




```
STEM               75.644639
No Major           15.847166
Humanities          3.492014
Other               1.988725
Business Degree     1.706859
Arts                1.320597
Name: major_discipline, dtype: float64
```

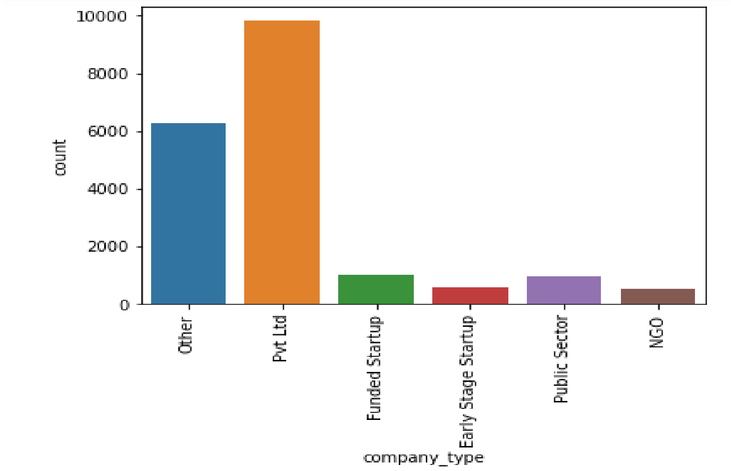
Company Size: Companies having employees in the range of 50-99 have maximum employees.

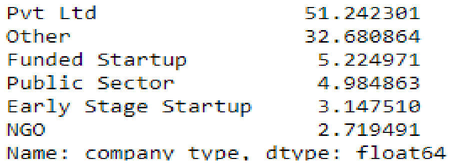




```
50-99        46.262658
100-500      13.555695
10000+       10.538678
49            7.678255
1000-4999     7.605178
<10           6.843094
500-999       4.577722
5000-9999     2.938720
Name: company_size, dtype: float64
```

Company Type: Maximum employees are working in the Private Limited Companies.

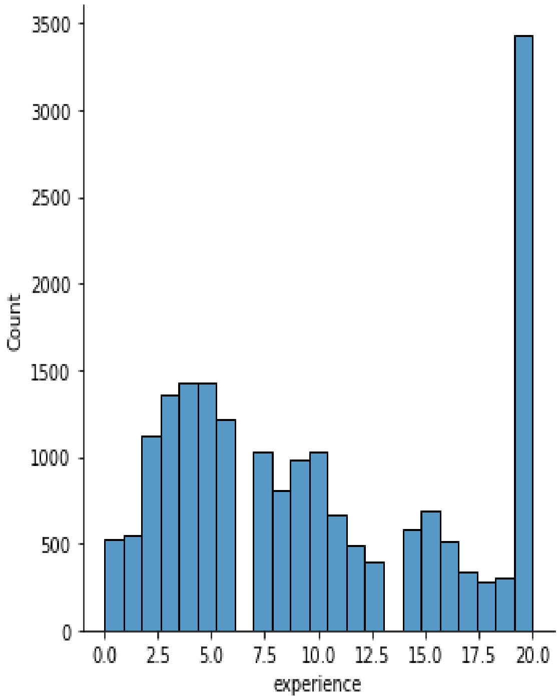


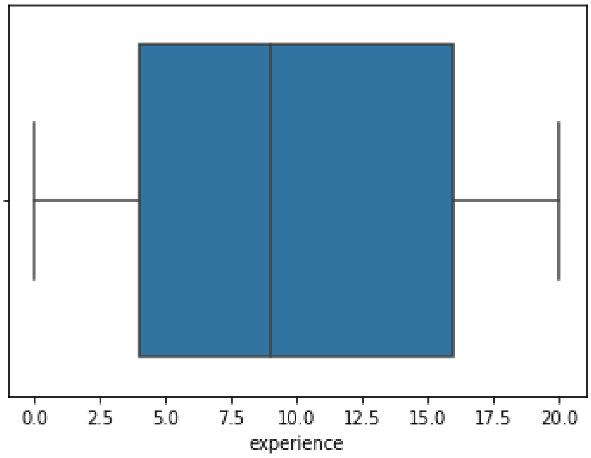


```
Pvt Ltd                 51.242301
Other                   32.680864
Funded Startup           5.224971
Public Sector            4.984863
Early Stage Startup      3.147510
NGO                      2.719491
Name: company type, dtype: float64
```

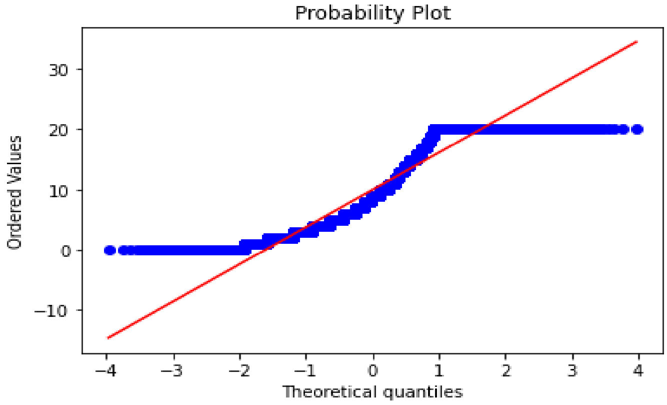
Numerical Variables

Experience: Most people are having experience of 20 years.

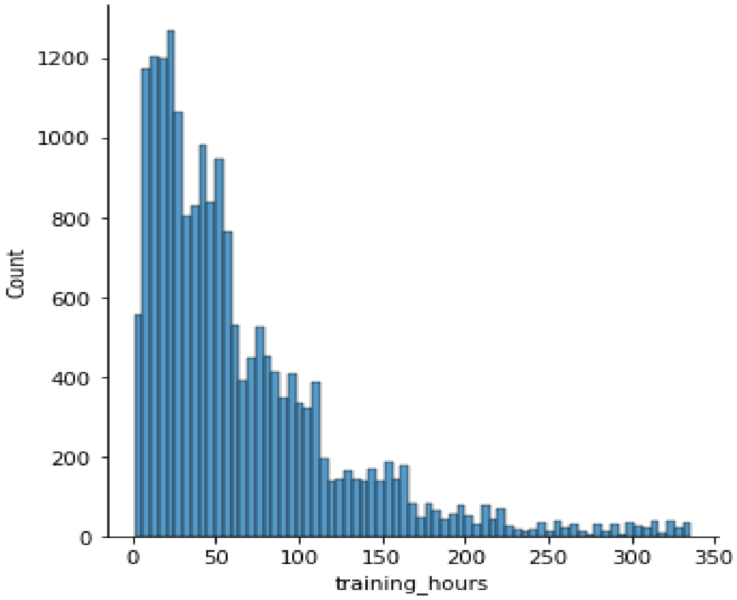


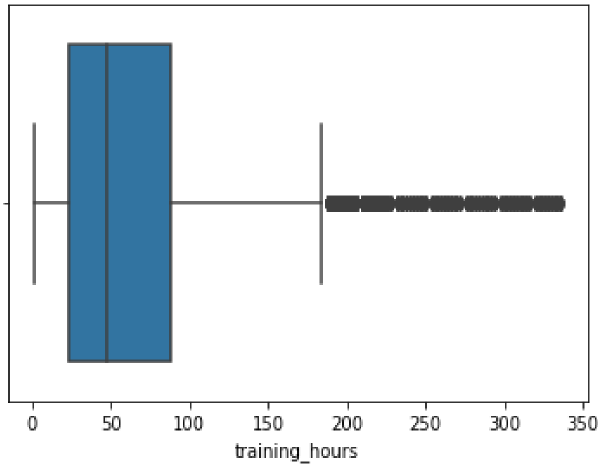


```
skewness of the  experience column is  0.3410292909488931
```

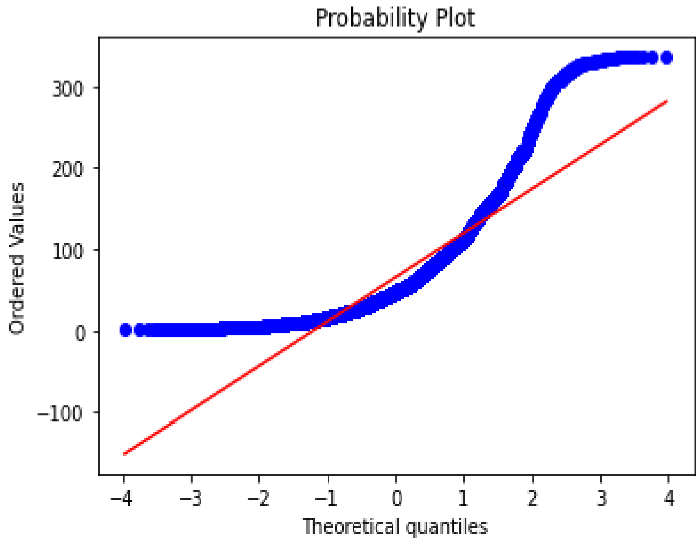


Training Hours: Most people have got trained a total of 10 to 100 hours.

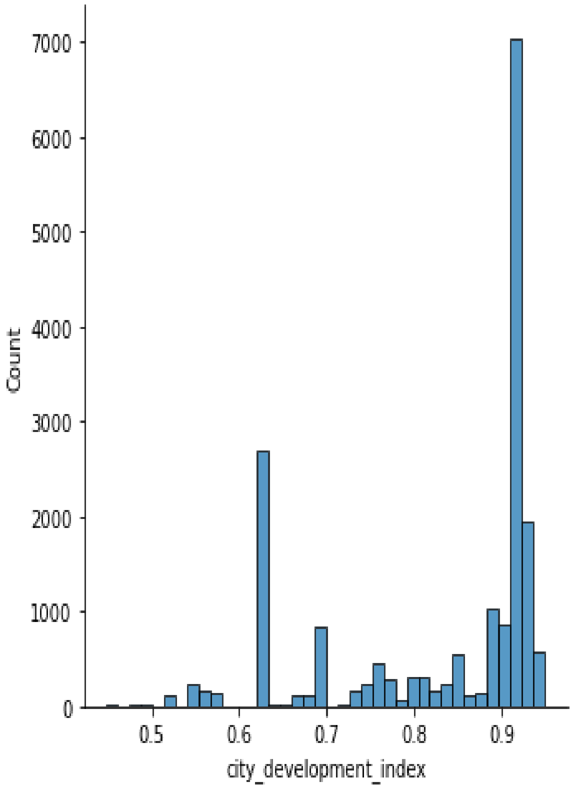


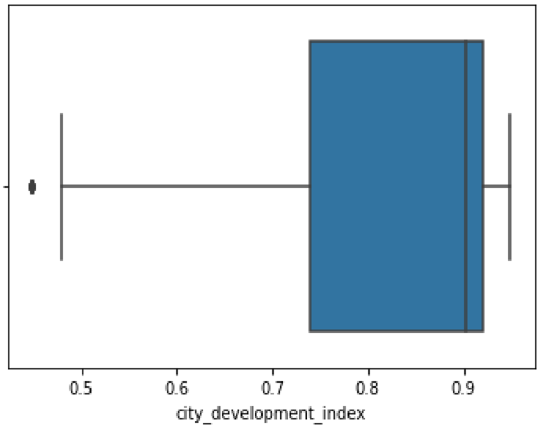


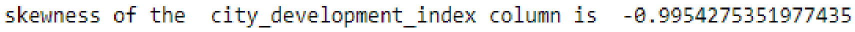
```
skewness of the  training_hours column is  1.8192372420221026
```


Probability Plot

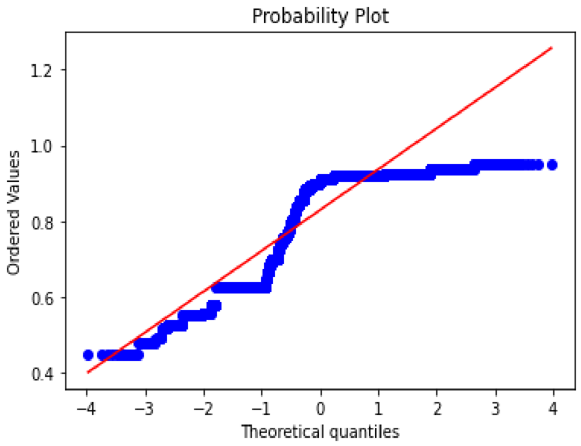
City Development Index: Most cities are having good development.







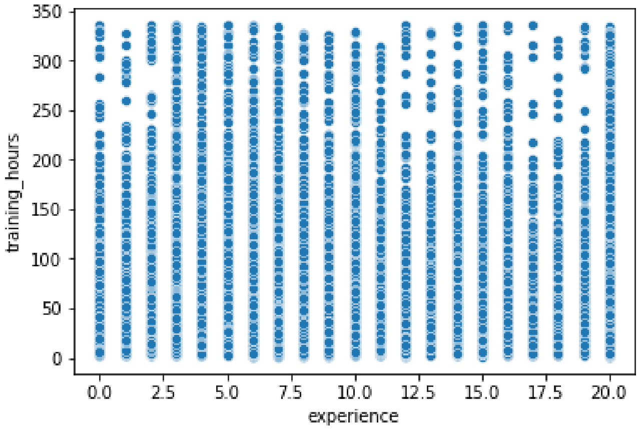
```
skewness of the  city_development_index column is  -0.9954275351977435
```



2.2 Bivariate Analysis

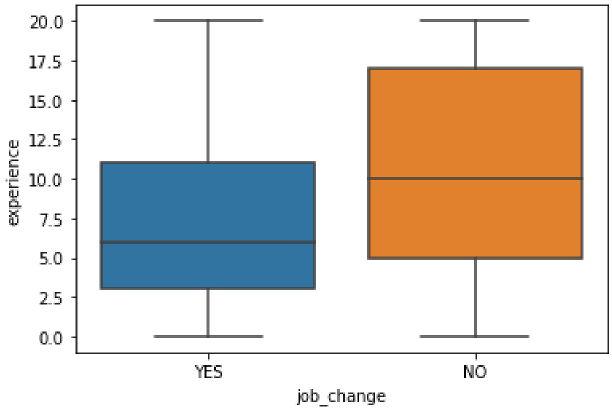
Numerical vs Numerical

Experience and Training_hours: People with high number of experience are trained less as compare to people with less experience.



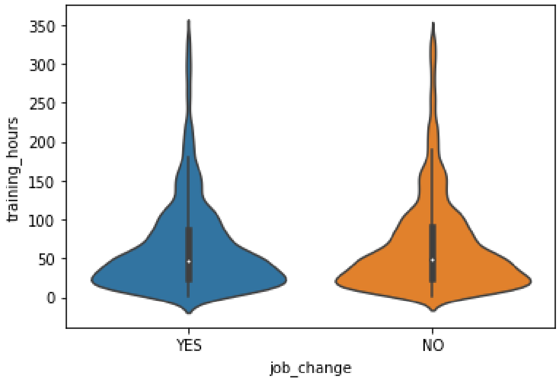
Numerical VS Categorical

Job Change and Experience: We can see that the median experience of who don't change jobs is around 10 as compared to 7 yes for the people who don't change jobs.

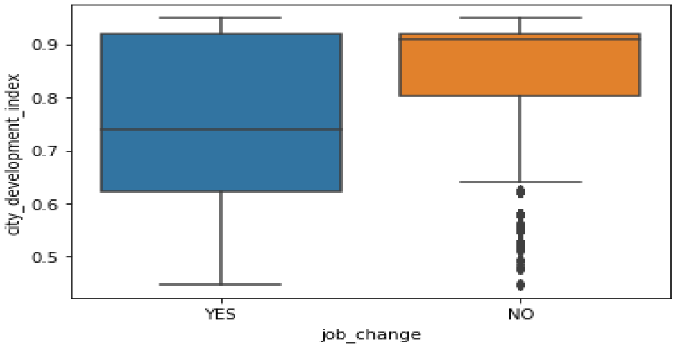


Job Change and Training Hours:

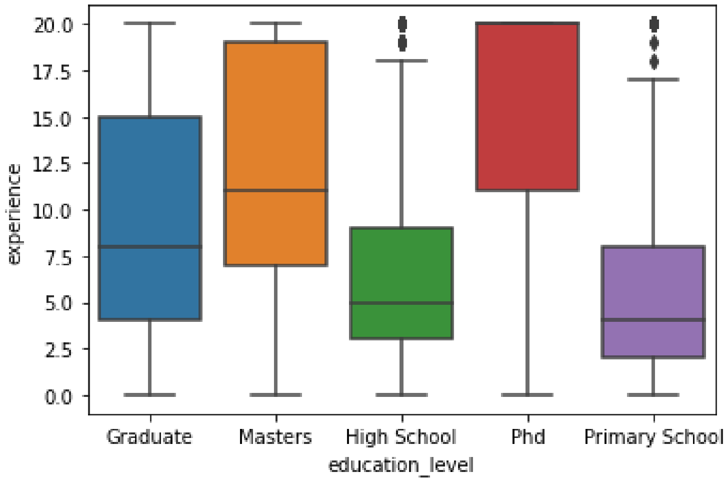
We can see that the people both who change jobs and don't change jobs are quite similar for both the people who change jobs and those who don't change jobs.



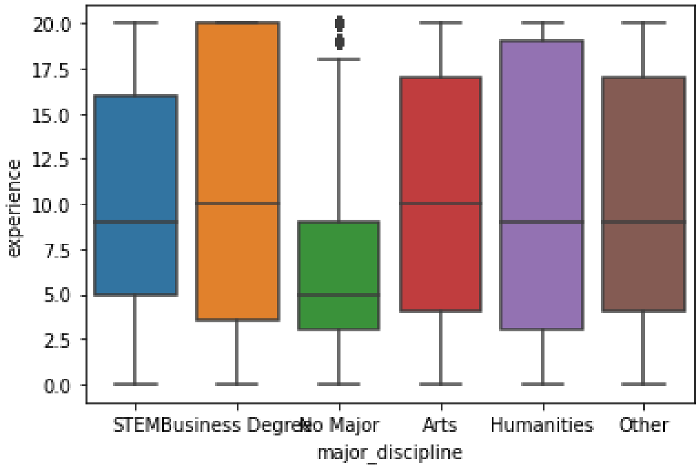
Job Change and City Development Index: We can see that the median of people who are unwilling to change jobs have high city deleopment index which can tell us that people living in metro cities are less prone to josb changes.



Job Change and Experience: The most high experience on average by phd as they have the highest qualtification we can only make out that the higher the education the higer the experience level

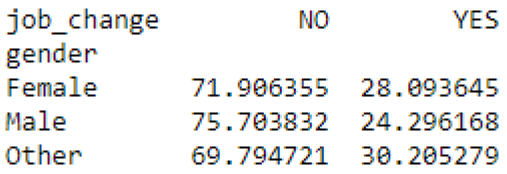


Major Discipline and Experience: Business degree and the people with no major have the highest experience level as compared to the rest

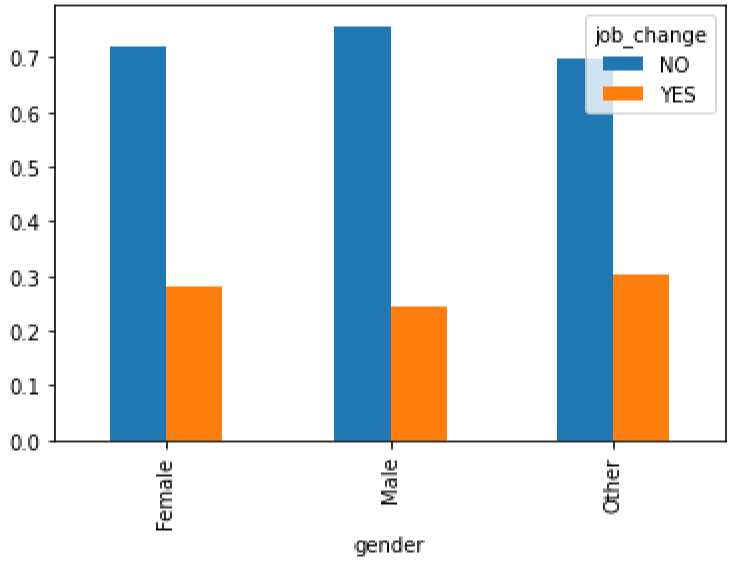


Categorical vs Categorical

Gender and Job Change:



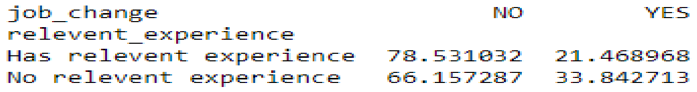
```
job_change         NO        YES
gender
Female      71.906355  28.093645
Male        75.703832  24.296168
Other       69.794721  30.205279
```



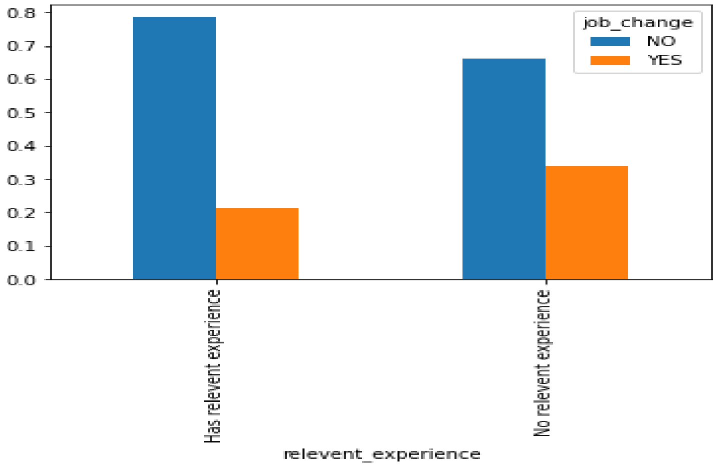
71.90 percent of females are not interested in job change and 28.10 perecnt are interested in job change
75.70 percent of males are not interested in job change and 24.30 perecnt are interested in job change
69.79 percent of gender named others are not interested in job change and 30.21 perecnt are interested in job change

Relevant Experience and Job Change:

78.53 percent people who hass relevent experience are not interested in job change and 21.47 precent peoples are interested in job change
66.15% percent people who has No relevent experience are not interested in job change and 33.85 precent peoples are interested in job change.



```
job_change                      NO        YES
relevent_experience
Has relevent experience  78.531032  21.468968
No relevent experience   66.157287  33.842713
```

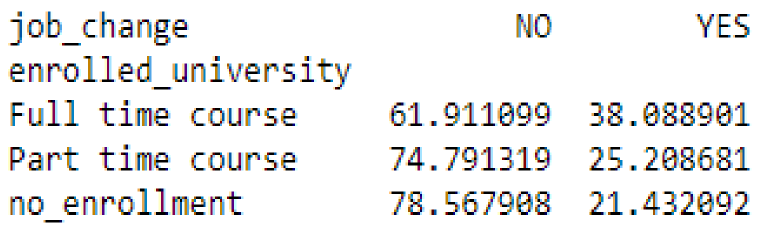


Enrolled University and Job Change:

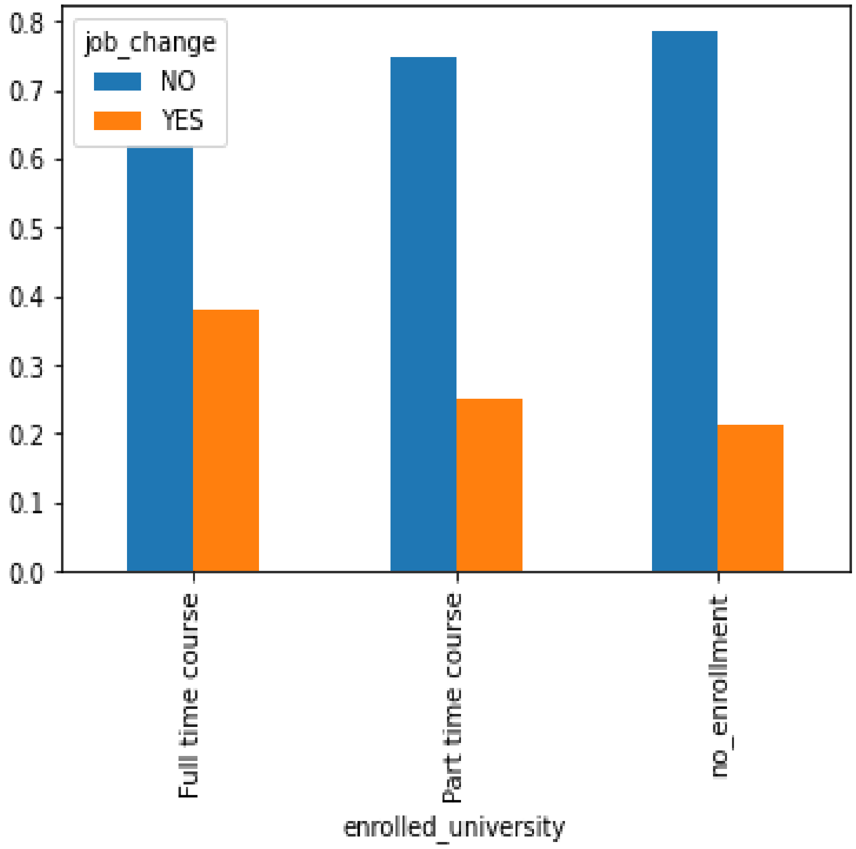
61.91 percent people with Full time course are not interested in job change and 38.09 precent peoples are interested in job change

74.79 percent people with Part time course are not interested in job change and 25.21 precent peoples are interested in job change

78.56 percent people with no_enrollment are not interested in job change and 21.44 precent peoples are interested in job change



```
job_change                    NO        YES
enrolled_university
Full time course       61.911099  38.088901
Part time course       74.791319  25.208681
no_enrollment          78.567908  21.432092
```



Major Discipline and Job Change:

People who has Arts as major discipline in which 79.05 percent are not interested and 20.95 percent are intersted in Job change

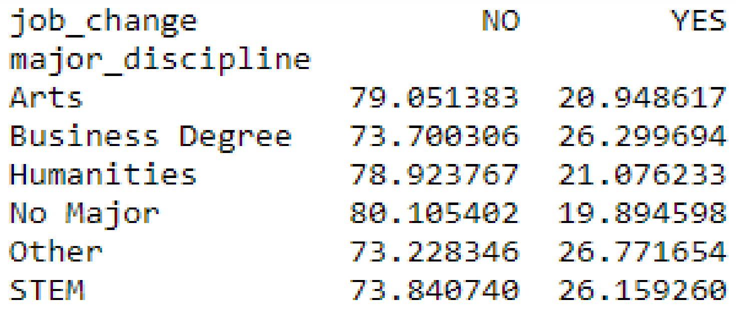
People who has Business Degree as major discipline in which 73.70 percent are not interested and 26.30 percent are intersted in Job change

People who has Humanities as major discipline in which 78.92 percent are not interested and 21.08 percent are intersted in Job change

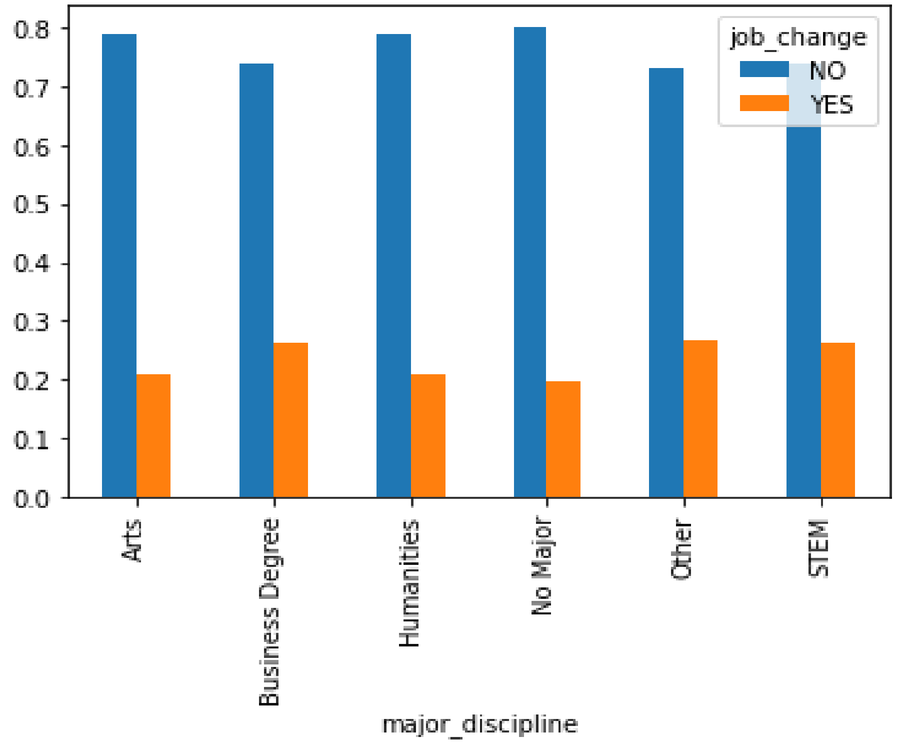
People who has No Major as major discipline in which 80.10 percent are not interested and 19.90 percent are intersted in Job change

People who has Other as major discipline in which 73.22 percent are not interested and 26.78 percent are intersted in Job change.

People who has STEM as major discipline in which 73.84 percent are not interested and 26.16 percent are intersted in Job change.



```
job_change              NO        YES
major_discipline
Arts              79.051383  20.948617
Business Degree   73.700306  26.299694
Humanities        78.923767  21.076233
No Major          80.105402  19.894598
Other             73.228346  26.771654
STEM              73.840740  26.159260
```



Education Level and Job Change:

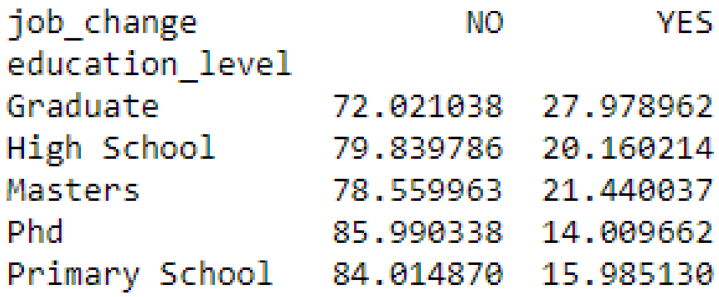
People who are Graduate in education level in which 72.03 percent are not interested and 27.97 percent are intersted in Job change

People who are qualified till High School in education level in which 79.83 percent are not interested and 20.17 percent are intersted in Job change

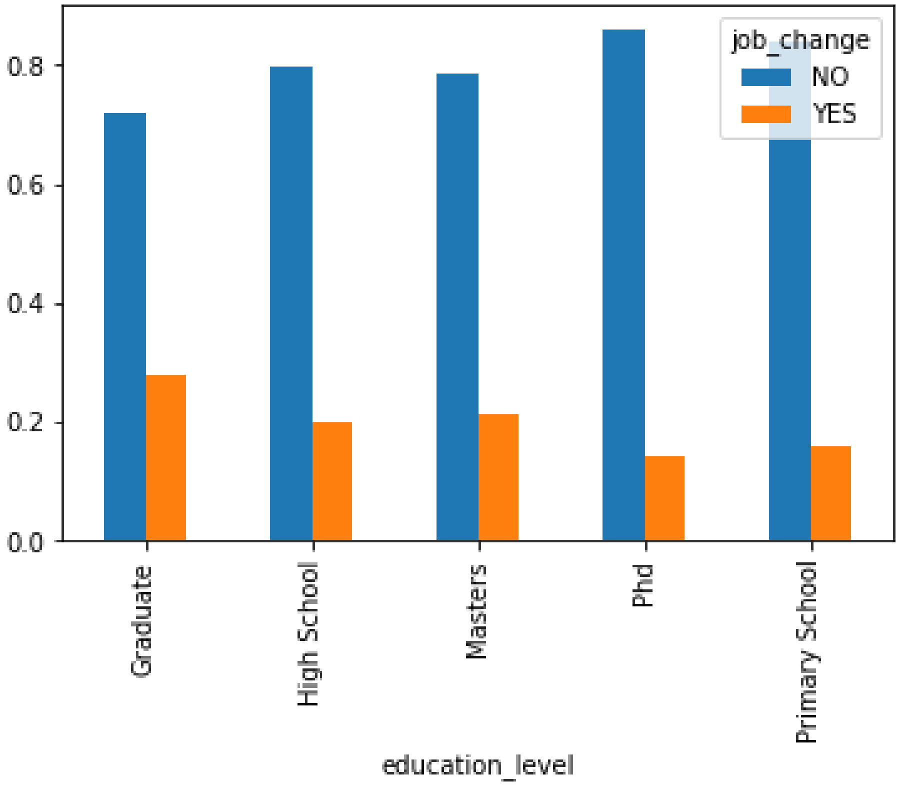
People who has the Masters Degree in education level in which 78.55 percent are not interested and 21.45 percent are intersted in Job change

People who are Phd Holder in education level in which 86 percent are not interested and 14 percent are intersted in Job change

People who are qualified till Primary School in education level in which 84 percent are not interested and 16 percent are intersted in Job change



```
job_change              NO        YES
education_level
Graduate         72.021038  27.978962
High School      79.839786  20.160214
Masters          78.559963  21.440037
Phd              85.990338  14.009662
Primary School   84.014870  15.985130
```



Company Size and Job Change:

People who are working in 100-500 large company size in which 83.63 percent are not interested and 16.37 percent are intersted in Job change

People who are working in 1000-4999 large company size in which 82.56 percent are not interested and 17.44 percent are intersted in Job change

People who are working in 10000+ large company size in which 80.93 percent are not interested and 19.07 percent are intersted in Job change

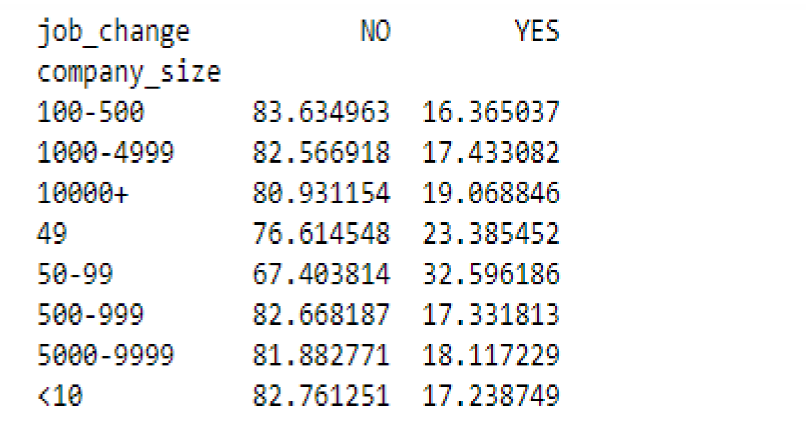
People who are working in 49 large company size in which 76.61 percent are not interested and 23.39 percent are intersted in Job change

People who are working in 50-99 large company size in which 67.40 percent are not interested and 32.60 percent are intersted in Job change

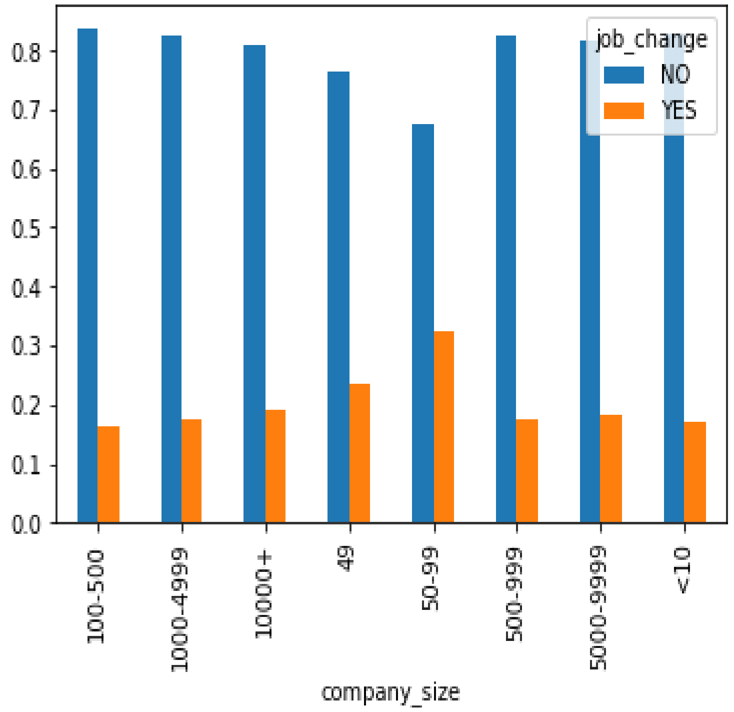
People who are working in 500-999 large company size in which 82.66 percent are not interested and 17.34 percent are intersted in Job change

People who are working in 5000-9999 large company size in which 81.88 percent are not interested and 11.12 percent are intersted in Job change

People who are working in <10 large company size in which 82.76 percent are not interested and 17.24 percent are intersted in Job change



```
job_change          NO        YES
company_size
100-500      83.634963  16.365037
1000-4999    82.566918  17.433082
10000+       80.931154  19.068846
49           76.614548  23.385452
50-99        67.403814  32.596186
500-999      82.668187  17.331813
5000-9999    81.882771  18.117229
<10          82.761251  17.238749
```



Company Type and Job Change:

People who are working in Early Stage Startup company type in which 76.45 percent are not interested and 23.55 percent are intersted in Job change

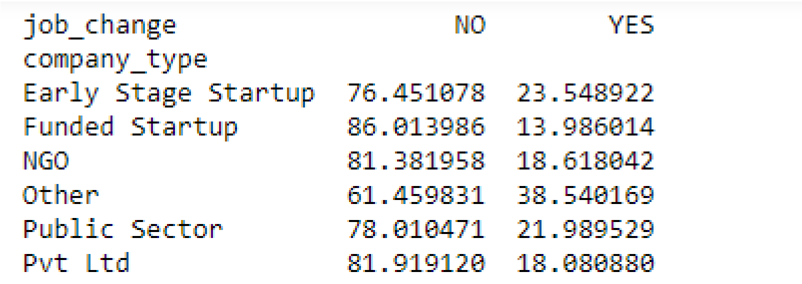
People who are working in Early Stage Startup company type in which 86 percent are not interested and 14 percent are intersted in Job change

People who are working in Early Stage Startup company type in which 81.38 percent are not interested and 11.62 percent are intersted in Job change

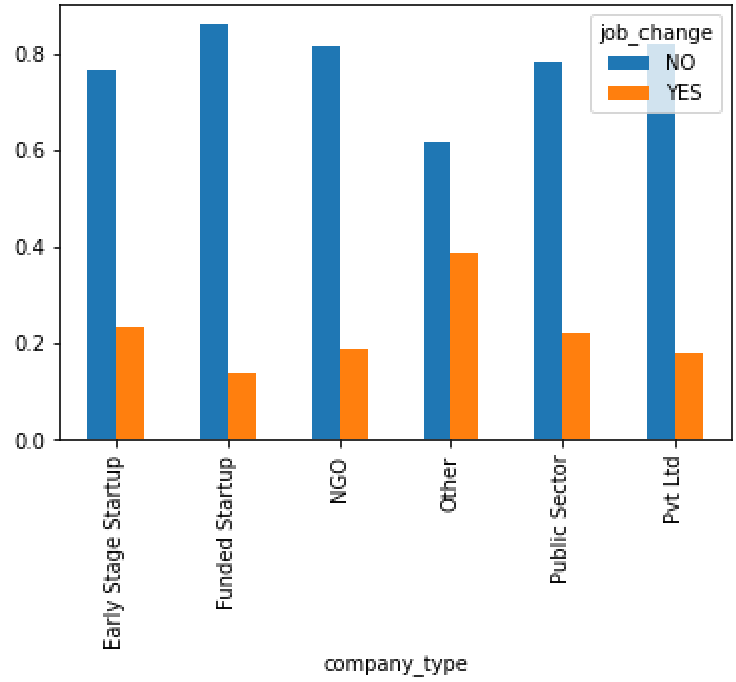
People who are working in Early Stage Startup company type in which 61.45 percent are not interested and 38.55 percent are intersted in Job change

People who are working in Early Stage Startup company type in which 78 percent are not interested and 22 percent are intersted in Job change

People who are working in Early Stage Startup company type in which 82 percent are not interested and 18 percent are intersted in Job change



```
job_change                  NO        YES
company_type
Early Stage Startup  76.451078  23.548922
Funded Startup       86.013986  13.986014
NGO                  81.381958  18.618042
Other                61.459831  38.540169
Public Sector        78.010471  21.989529
Pvt Ltd              81.919120  18.080880
```



Last New Job and Job Change:

People who changed their jobs 1 time in which 74 percent are not interested and 26 percent are intersted in Job change

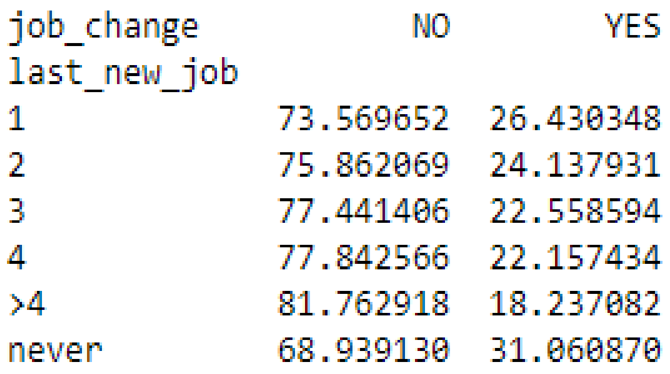
People who changed their jobs 2 time in which 76 percent are not interested and 24 percent are intersted in Job change

People who changed their jobs 3 time in which 77 percent are not interested and 23 percent are intersted in Job change

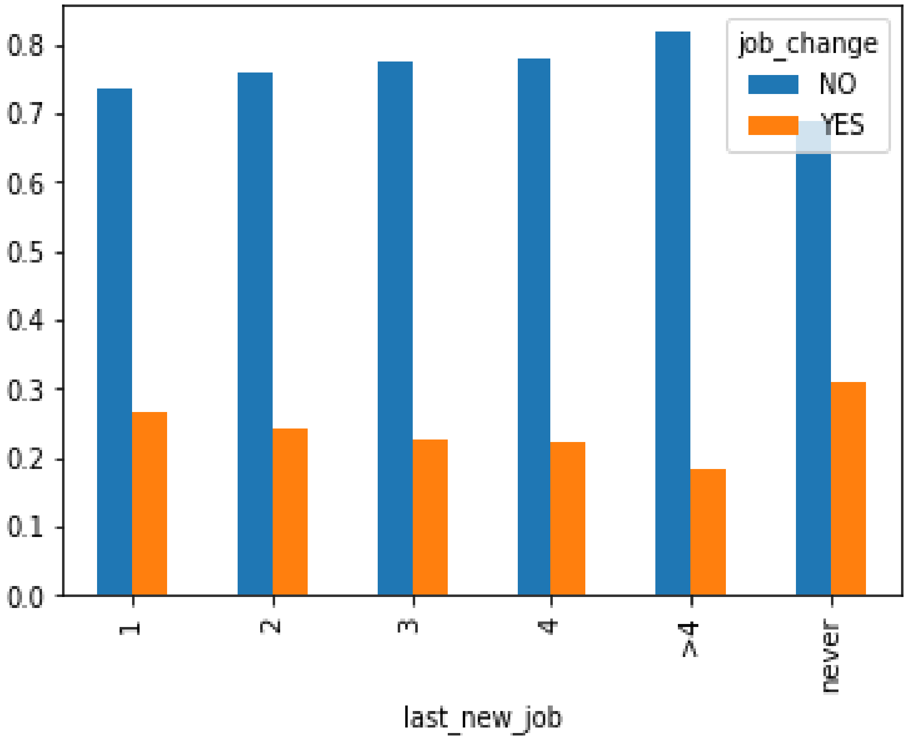
People who changed their jobs 4 time in which 78 percent are not interested and 22 percent are intersted in Job change

People who changed their jobs >4 time in which 82 percent are not interested and 18 percent are intersted in Job change

People who never changed their jobs time in which 69 percent are not interested and 31 percent are intersted in Job change



```
job_change           NO        YES
last_new_job
1             73.569652  26.430348
2             75.862069  24.137931
3             77.441406  22.558594
4             77.842566  22.157434
>4            81.762918  18.237082
never         68.939130  31.060870
```



2.3 Correlation of the Dataset

Now we have removed all duplicated values, outliers, and irrelevant data. The correlation ofthe full dataset is as below:

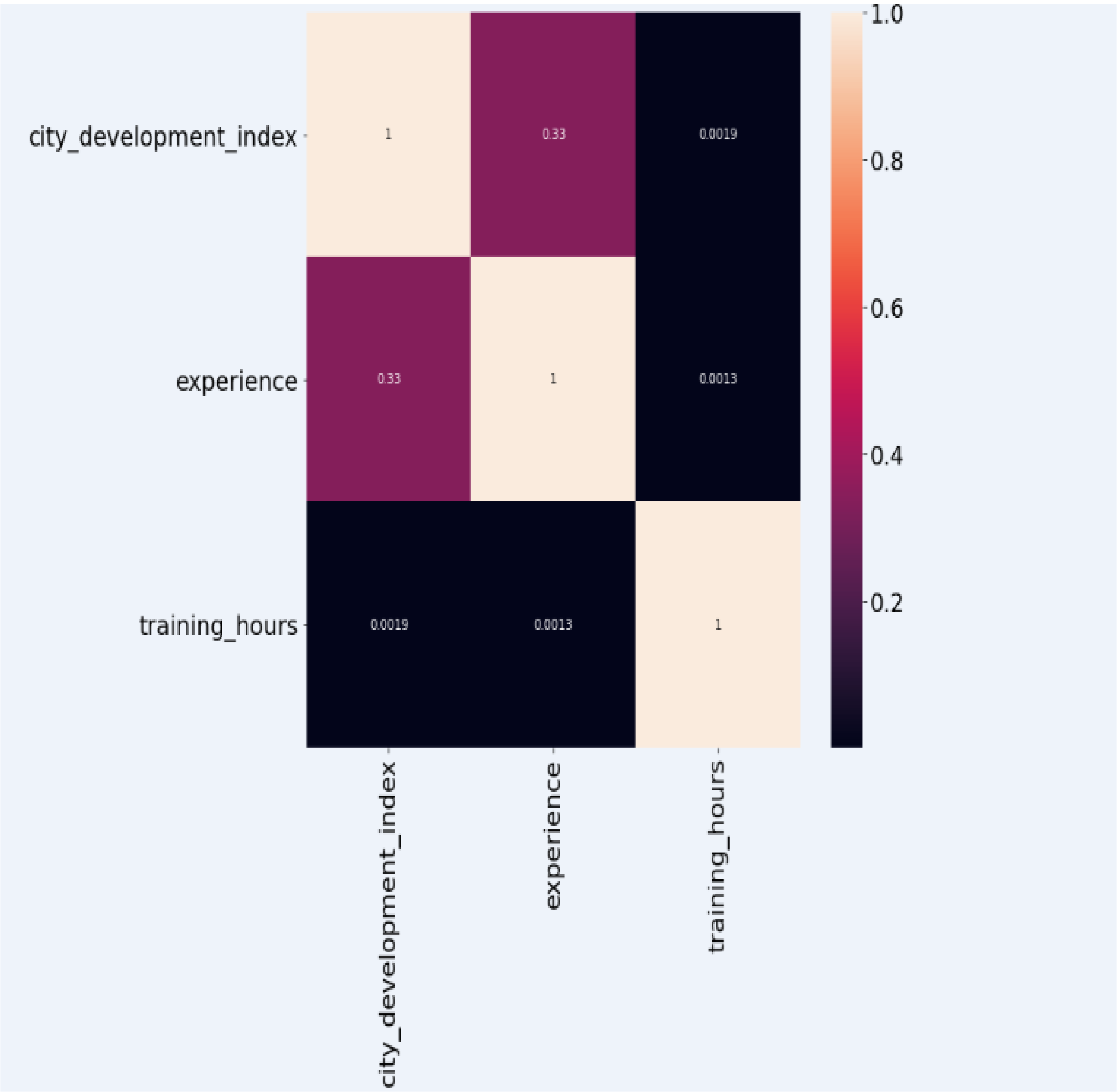


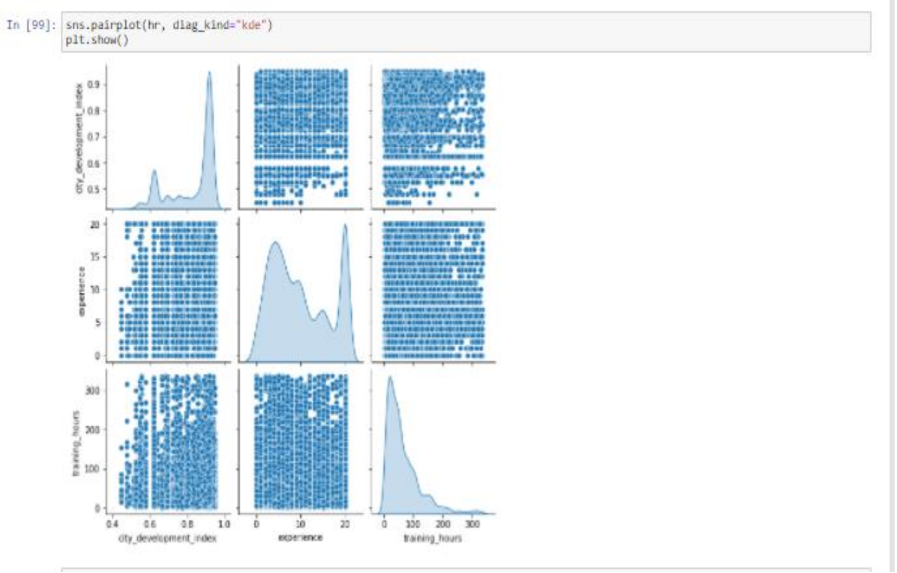
Figure: There is weak correlation observed between among dependent features

3. STATISTICAL ANALYSIS:

Statistics refers to the study of collecting, analyzing, interpreting, presenting, and organizing data in a particular manner.

3.1 PAIRPLOT

1. Firstly, we have made a pair plot visualization to analyze the relationships the features are trying to explain and make some inferences out of it as to which might be useful features and to others who might just be the noise in the final data modeling.

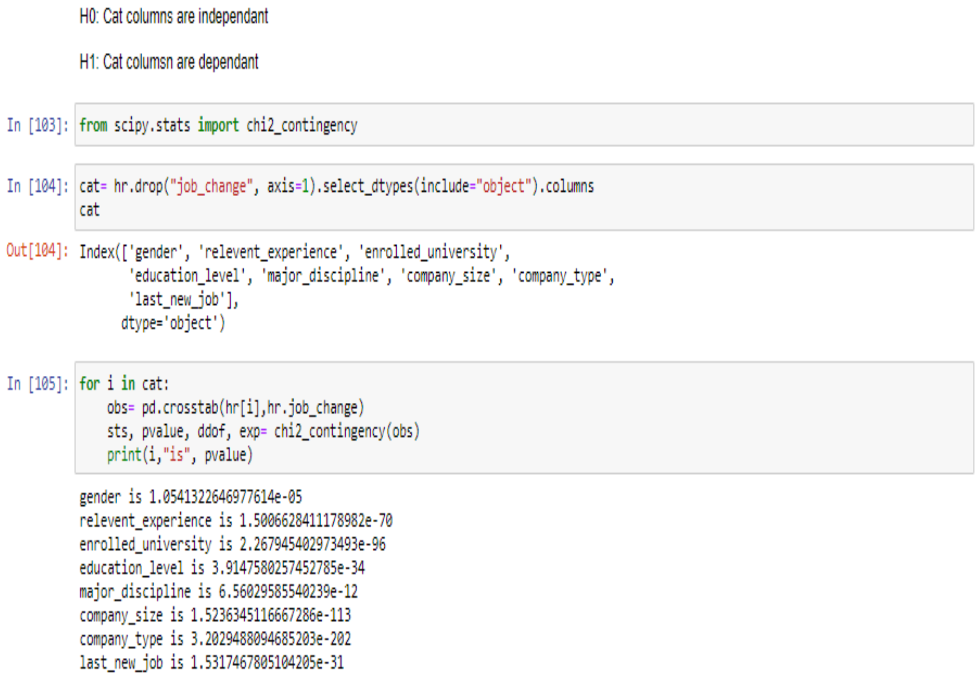


```
In [99]: sns.pairplot(hr, diag_kind="kde")
         plt.show()
```

3.2 CHI SQUARE CONTINGENCY TEST FOR INDEPEDNANT ATTRIBUTES

2. Secondly, we have done the chi squared test to compare the observed results with expected results and we concluded:

i. All the categorical columns are independent of each other.



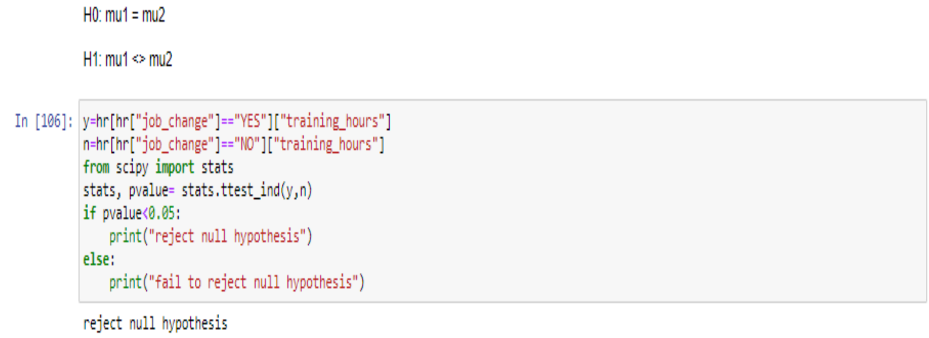
H0: Cat columns are independant

H1: Cat columsn are dependant

```
In [103]: from scipy.stats import chi2_contingency

In [104]: cat= hr.drop("job_change", axis=1).select_dtypes(include="object").columns
          cat

Out[104]: Index(['gender', 'relevent_experience', 'enrolled_university',
                 'education_level', 'major_discipline', 'company_size', 'company_type',
                 'last_new_job'],
                dtype='object')

In [105]: for i in cat:
              obs= pd.crosstab(hr[i],hr.job_change)
              sts, pvalue, ddof, exp= chi2_contingency(obs)
              print(i,"is", pvalue)

          gender is 1.0541322646977614e-05
          relevent_experience is 1.5006628411178982e-70
          enrolled_university is 2.267945402973493e-96
          education_level is 3.9147580257452785e-34
          major_discipline is 6.56029585540239e-12
          company_size is 1.5236345116667286e-113
          company_type is 3.2029488094685203e-202
          last_new_job is 1.5317467805104205e-31
```

3.3 TWO SAMLE TTEST ON TRAINING HOURS WITH RESPECT TO JOB CHANGE

iii. The mean training hours of employees who change jobs is different than those who don’t change jobs.



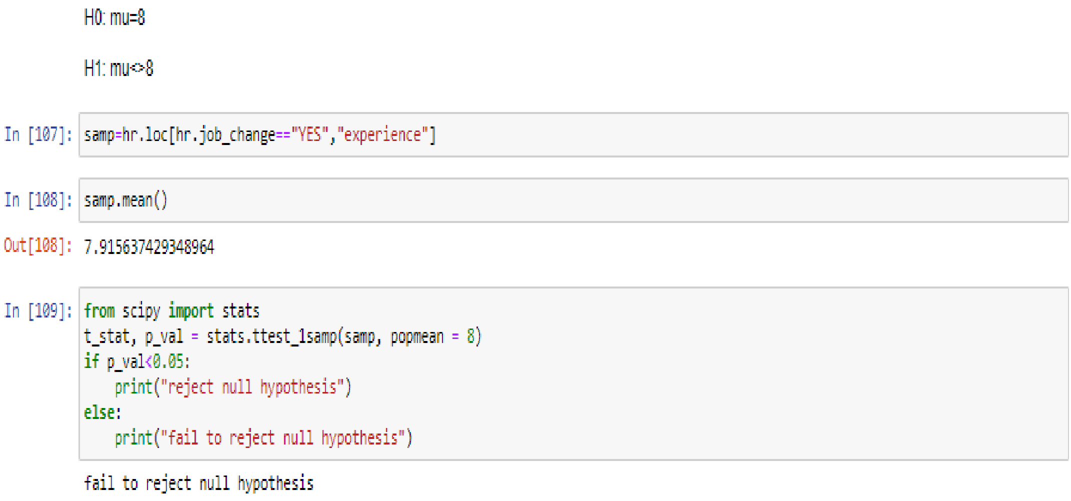
H0: mu1 = mu2

H1: mu1 <> mu2

```
In [106]: y=hr[hr["job_change"]=="YES"]["training_hours"]
          n=hr[hr["job_change"]=="NO"]["training_hours"]
          from scipy import stats
          stats, pvalue= stats.ttest_ind(y,n)
          if pvalue<0.05:
              print("reject null hypothesis")
          else:
              print("fail to reject null hypothesis")

          reject null hypothesis
```

3.4 ONCE SAMPLE TTEST ON TRAINING HOURS

iii. . The Avg Experience of people who change jobs is not 8yrs as opposed to the real avg experience of employees.



H0: mu=8

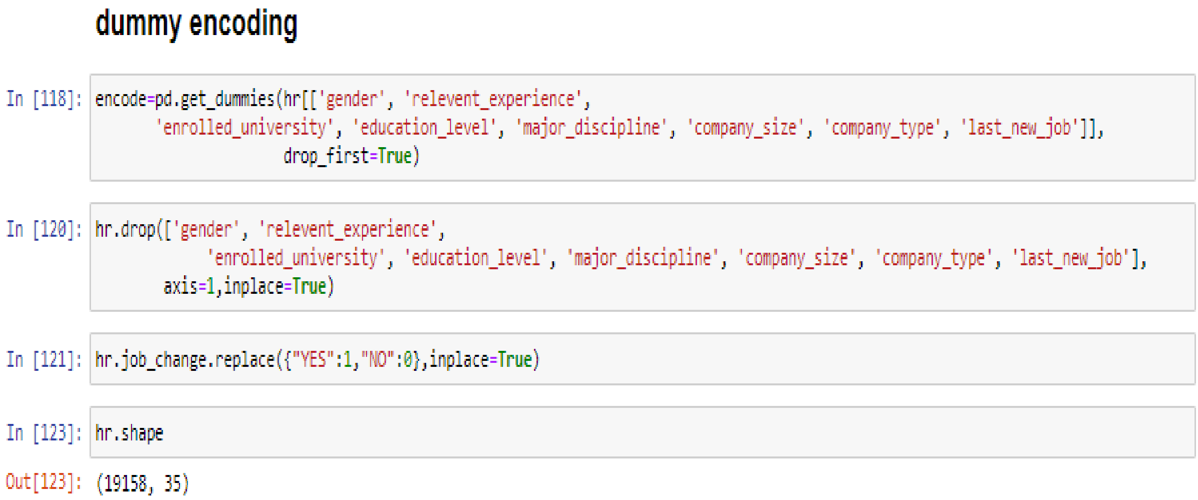
H1: mu<>8

```
In [107]: samp=hr.loc[hr.job_change=="YES","experience"]

In [108]: samp.mean()

Out[108]: 7.915637429348964

In [109]: from scipy import stats
          t_stat, p_val = stats.ttest_1samp(samp, popmean = 8)
          if p_val<0.05:
              print("reject null hypothesis")
          else:
              print("fail to reject null hypothesis")

          fail to reject null hypothesis
```

4.FEATURE ENGINNERING

Feature engineering is a process of transforming the given data into a form which is easier to interpret by the machine learning model.

For the features to be in a proper format that can be interpreted by the machine learning algorithms we have applied the dummy encoding on the categorical columns of the dataset. Dummy encoding basically transforms the categorical variable into a set of binary variables which is interpretable in the form of True/False by the model.



dummy encoding

```
In [118]: encode=pd.get_dummies(hr[['gender', 'relevent_experience',
                 'enrolled_university', 'education_level', 'major_discipline', 'company_size', 'company_type', 'last_new_job']],
                              drop_first=True)

In [120]: hr.drop(['gender', 'relevent_experience',
                      'enrolled_university', 'education_level', 'major_discipline', 'company_size', 'company_type', 'last_new_job'],
                   axis=1,inplace=True)

In [121]: hr.job_change.replace({"YES":1,"NO":0},inplace=True)

In [123]: hr.shape

Out[123]: (19158, 35)
```

5. MACHINE LEARNING/MODEL BUILDING :

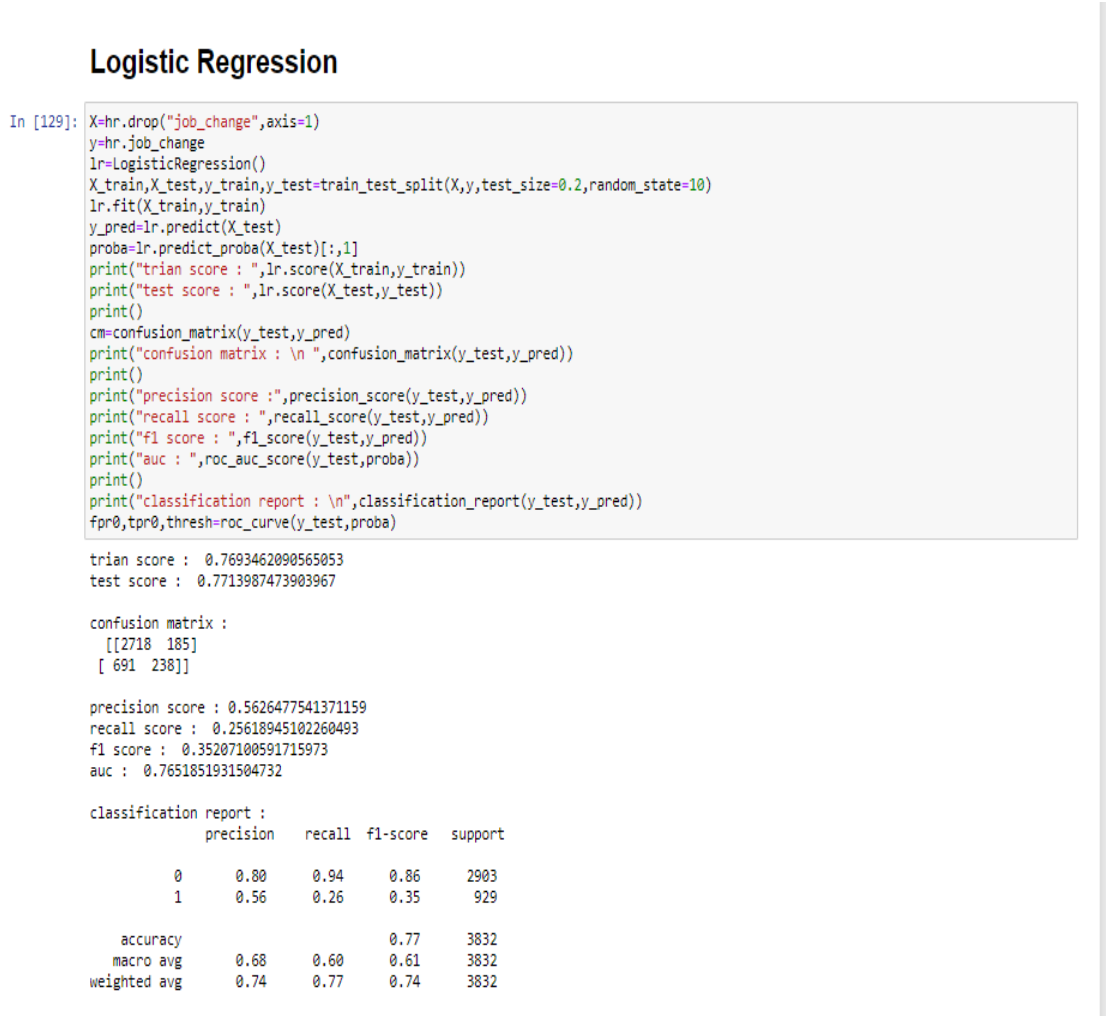
Machine learning is the study of computer algorithms that improve automatically through experience and by the use of data.

5.1 BASE MODEL :

We have taken the Logistic Regression model as our base learner model for this project.

Reason for base model:

We chose logistic regression as our base model since it is a predictive technique that uses independent factors to predict the dependent variable, similar to Linear Regression, except the dependent variable must be categorical. It is based on two fundamental assumptions: no outliers and no multicollinearity. We dealt with outliers in our data, and there is no multicollinearity. Essentially, it employs the Logistic function to model the conditional probability and calculate the conditional probability of the dependent variables Y and X by 'P(Y=1|X) or P(Y=0|X),' where P(Y|X) is approximated as a sigmoid function.



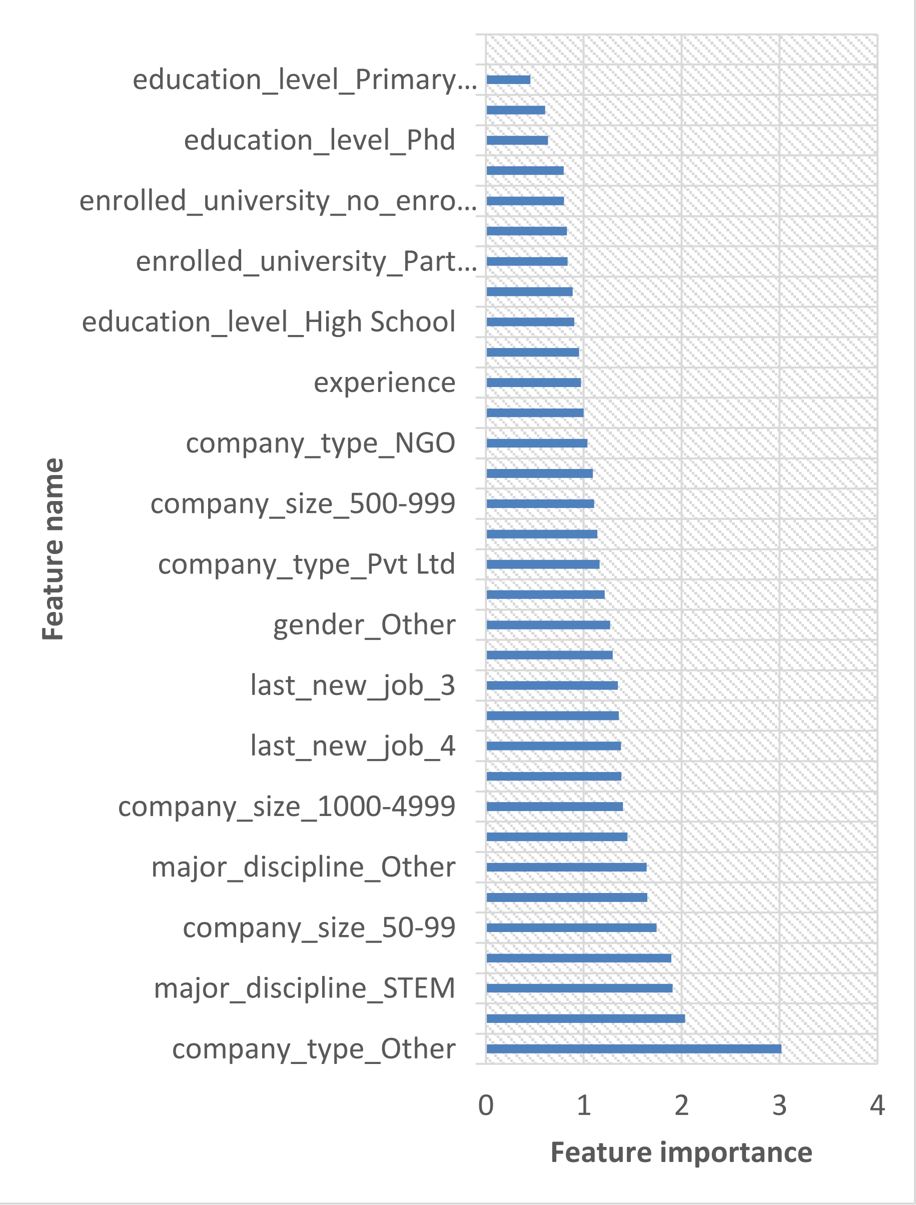
Logistic Regression

```
In [129]: X=hr.drop("job_change",axis=1)
          y=hr.job_change
          lr=LogisticRegression()
          X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=10)
          lr.fit(X_train,y_train)
          y_pred=lr.predict(X_test)
          proba=lr.predict_proba(X_test)[:,1]
          print("trian score : ",lr.score(X_train,y_train))
          print("test score : ",lr.score(X_test,y_test))
          print()
          cm=confusion_matrix(y_test,y_pred)
          print("confusion matrix : \n ",confusion_matrix(y_test,y_pred))
          print()
          print("precision score :",precision_score(y_test,y_pred))
          print("recall score : ",recall_score(y_test,y_pred))
          print("f1 score : ",f1_score(y_test,y_pred))
          print("auc : ",roc_auc_score(y_test,proba))
          print()
          print("classification report : \n",classification_report(y_test,y_pred))
          fpr0,tpr0,thresh=roc_curve(y_test,proba)
```

```
trian score :  0.7693462090565053
test score :  0.7713987473903967

confusion matrix :
  [[2718  185]
 [ 691  238]]

precision score : 0.5626477541371159
recall score :  0.25618945102260493
f1 score :  0.35207100591715973
auc :  0.7651851931504732

classification report :
               precision    recall  f1-score   support

           0       0.80      0.94      0.86      2903
           1       0.56      0.26      0.35       929

    accuracy                           0.77      3832
   macro avg       0.68      0.60      0.61      3832
weighted avg       0.74      0.77      0.74      3832
```

From the above results we can see that the Logistic Regression model is performing quite ok is we consider the train and test scores but as we go on to analyze the precision ,recall and f1-scores we can find that it is doing a terrible job at classification as we can even see high False Negatives.

Feature Importance is a number assigned to the features of a Machine Learning model that defines how "essential" a feature is to the model's prediction. It can aid in feature selection and provide highly important insights into our data. Based on the relevant feature importance analysis, we can conclude that the company type _other has the most value among all the other characteristics.



Model Interpretation:

We will be evaluating our model using classification measures such as Precision Score, Recall Score, and F1 Score. Because Precision is the ratio of successfully predicted positive observations to total predicted positive observations, Recall is the ratio of correctly predicted positive observations to total observed positive observations, and F1 Score is the weighted average of Precision and Recall. Because our Precision score is less than 0.5 and our Recall score is less than 0.5, we may conclude that Logistic Regression is not doing well as a model for our data. So, in the near future, we will try to add new models and see if our results improve or not.

5.2 CLASSIFIER COMPARISON;

1. Logistic Regression –

Advantages of Logistics Regression :

- Logistic Regression Are very easy to understand
- It requires less training
- Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.

Disadvantages of Logistics Regression :

- Sometimes Lot of Feature Engineering Is required
- If the independent features are correlated it may affect performance
- It is often quite prone to noise and overfitting

❖ Feature Scaling is required

❖ Sensitive to missing values

❖ Performance Metrics Classification : Confusion Matrix,Precision,Recall, F1 score

❖ Impact of outliers : Like linear regression, estimates of the logistic regression are sensitive to the unusual observations: outliers, high leverage, and influential observations. Numerical examples and analysis are presented to demonstrate the most recent outlier diagnostic methods using data sets from medical domain

2. Decision Tree Classifier –

Advantages of Decision Tree :

- Clear Visualization: The algorithm is simple to understand, interpret and visualize as the idea is mostly used in our daily lives. Output of a Decision Tree can be easily interpreted by humans.
- Simple and easy to understand: Decision Tree looks like simple if-else statements which are very easy to understand.
- Decision Tree can be used for both classification and regression problems.
- Decision Tree can handle both continuous and categorical variables.

Disadvantages of Decision Tree :

- Overfitting: This is the main problem of the Decision Tree. It generally leads to overfitting of the data which ultimately leads to wrong predictions. In order to fit the data (even noisy data), it keeps generating new nodes and ultimately the tree becomes too complex to interpret. In this way, it loses its generalization capabilities. It performs very well on the trained data but starts making a lot of mistakes on the unseen data.
- High variance: As mentioned in point 1, Decision Tree generally leads to the overfitting of data. Due to the overfitting, there are very high chances of high variance in the output which leads to many errors in the final estimation and shows high inaccuracy in the results. In order to achieve zero bias (overfitting), it leads to high variance.
- Unstable: Adding a new data point can lead to re-generation of the overall tree and all nodes need to be recalculated and recreated.

❖ Feature Scaling is not required.

❖ Impact of outliers **:** It is not sensitive to outliers.Since, extreme values or outliers, never cause much reduction in RSS, they are never involved in split. Hence, tree based methods are insensitive to outliers

3.Random Forest Classifier –

Advantages of Random Forest Classifier :

- Doesn't Overfit
- Favourite algorithm for Kaggle competition
- Less Parameter Tuning required
- Decision Tree can handle both continuous and categorical variables.
- No feature scaling required: No feature scaling (standardization and normalization) required in case of Random Forest as it uses Decision Tree internally
- Suitable for any kind of ML problems.

Disadvantages of Random Forest Classifier :

- Biased With features having many categories
- Biased in multiclass classification problems towards more frequent classes.

❖ Feature Scaling is not required.
❖ Impact of outliers *:* Robust to Outliers

4. Adaboost –

Advantages of Adaboost :

- Doesn't Overfit
- It has few parameters to tune

Disadvantages of Adaboost :

- AdaBoost is extremely sensitive to Noisy data and outliers so if we use AdaBoost then it is highly recommended to eliminate them.

❖ Adaboost can handle missing values
❖ Feature Scaling is not required
❖ Impact of Outliers : Sensitive to outliers

5. Gradient Boost-

Advantages of Gradient Boost :

- It has a great performance
- It can solve complex non linear functions
- It is better in solve any kind of ML use cases
-

Disadvantages of Gradient Boost :

- It requires some amount of parameter tuning

❖ Cannot handle missing values.
❖ Feature Scaling is not required.
❖ Impact of Outliers : Robust to Outliers.

Impact to Outliers : Sensitive to outliers

6. KNN (K Nearest Neighbours) –

Advantages of KNN :

- ▪ Quick calculation time.
- ▪ Simple algorithm – to interpret.
- ▪ Versatile – useful for regression and classification.
- ▪ High accuracy

Disadvantages of KNN :

- ▪ No Training Period
- ▪ Lazy Learner
- ❖ Can handle missing values
- ❖ Feature Scaling is required.
- ❖ Impact to Outliers : Sensitive to Outliers

7. Naïve Bayes –

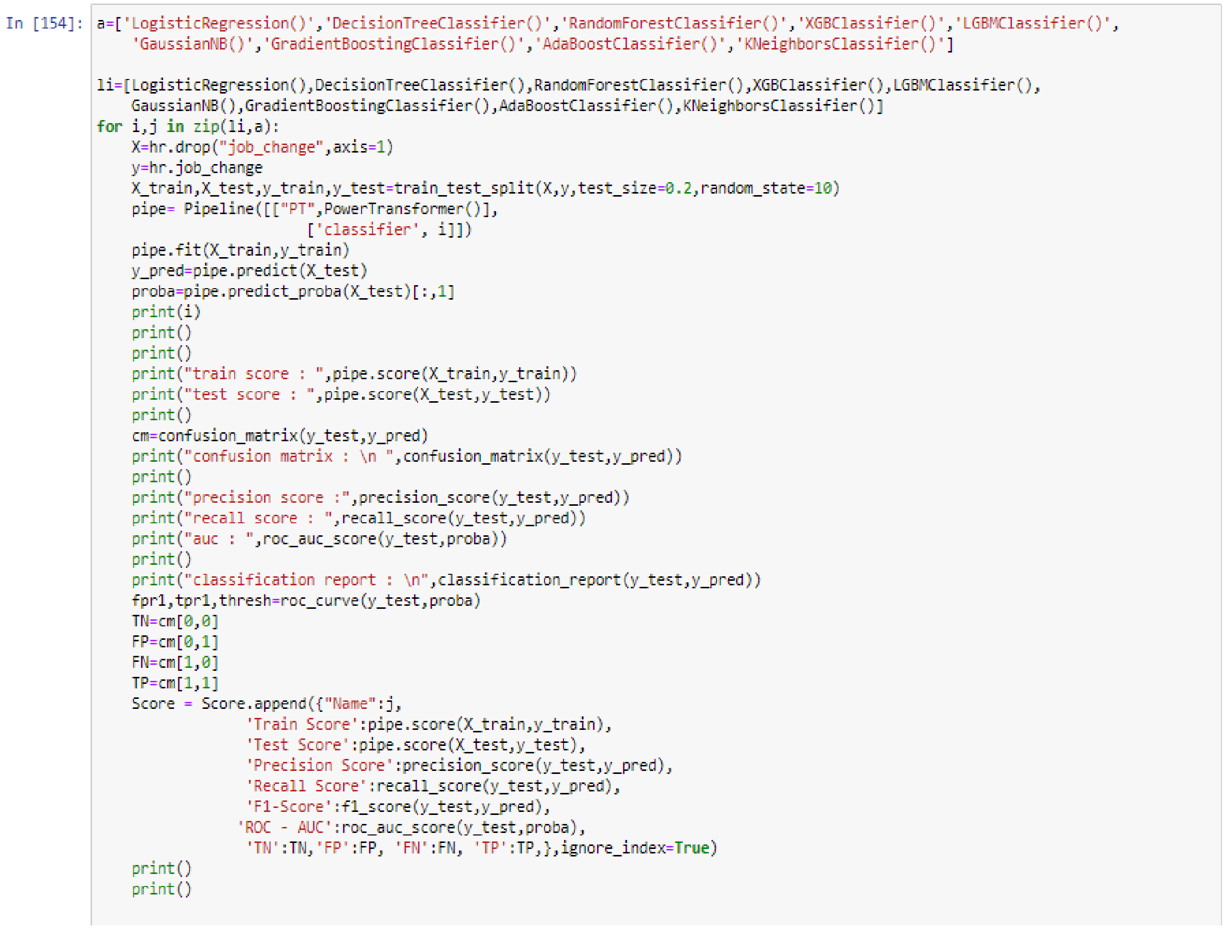
Advantages of Naïve Bayes :

- ▪ It is simple and easy to implement.
- ▪ It doesn't require as much training data.
- ▪ It handles both continuous and discrete data.
- ▪ It is highly scalable with the number of predictors and data points.

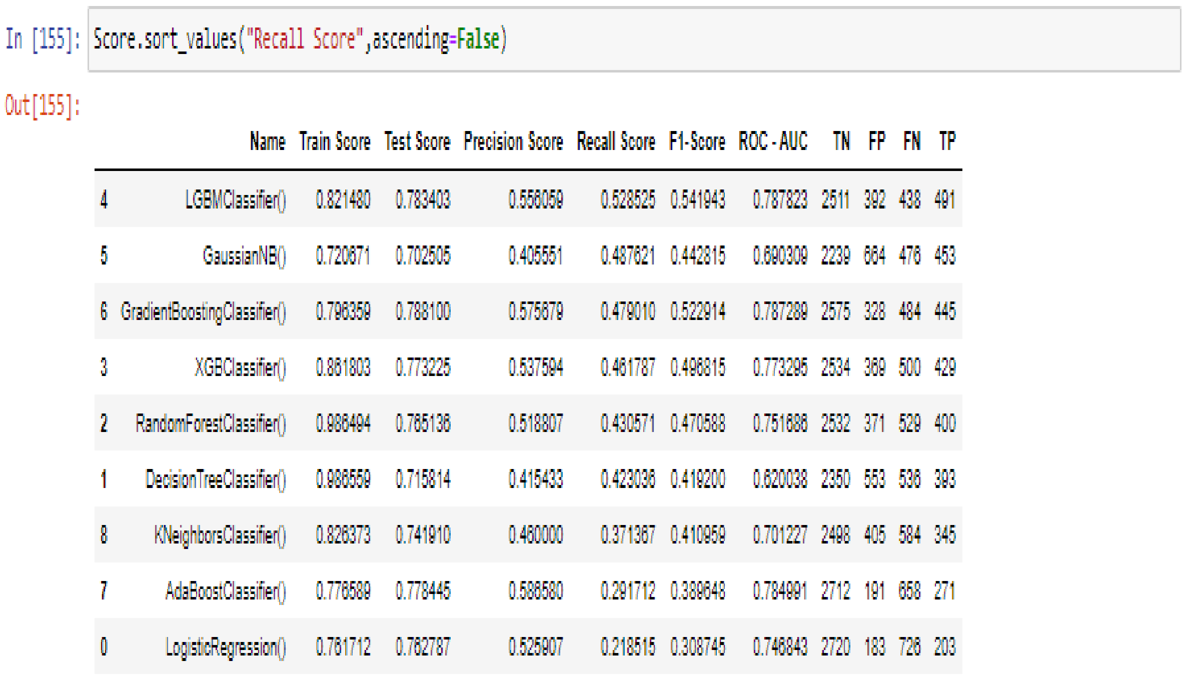
Disadvantages of Naïve Bayes :

- ▪ assumption of independent predictor features.
- ▪ bad estimator

- ❖ Can handle missing values
- ❖ Feature Scaling is not required
- ❖ Impact to Outliers : Robust to Outliers.



```
In [154]: a=['LogisticRegression()','DecisionTreeClassifier()','RandomForestClassifier()','XGBClassifier()','LGBMClassifier()',
              'GaussianNB()','GradientBoostingClassifier()','AdaBoostClassifier()','KNeighborsClassifier()']

          li=[LogisticRegression(),DecisionTreeClassifier(),RandomForestClassifier(),XGBClassifier(),LGBMClassifier(),
              GaussianNB(),GradientBoostingClassifier(),AdaBoostClassifier(),KNeighborsClassifier()]
          for i,j in zip(li,a):
              X=hr.drop("job_change",axis=1)
              y=hr.job_change
              X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=10)
              pipe= Pipeline([["PT",PowerTransformer()],
                                  ['classifier', i]])
              pipe.fit(X_train,y_train)
              y_pred=pipe.predict(X_test)
              proba=pipe.predict_proba(X_test)[:,1]
              print(i)
              print()
              print()
              print("train score : ",pipe.score(X_train,y_train))
              print("test score : ",pipe.score(X_test,y_test))
              print()
              cm=confusion_matrix(y_test,y_pred)
              print("confusion matrix : \n ",confusion_matrix(y_test,y_pred))
              print()
              print("precision score :",precision_score(y_test,y_pred))
              print("recall score : ",recall_score(y_test,y_pred))
              print("auc : ",roc_auc_score(y_test,proba))
              print()
              print("classification report : \n",classification_report(y_test,y_pred))
              fpr1,tpr1,thresh=roc_curve(y_test,proba)
              TN=cm[0,0]
              FP=cm[0,1]
              FN=cm[1,0]
              TP=cm[1,1]
              Score = Score.append({"Name":j,
                             'Train Score':pipe.score(X_train,y_train),
                             'Test Score':pipe.score(X_test,y_test),
                             'Precision Score':precision_score(y_test,y_pred),
                             'Recall Score':recall_score(y_test,y_pred),
                             'F1-Score':f1_score(y_test,y_pred),
                            'ROC - AUC':roc_auc_score(y_test,proba),
                             'TN':TN,'FP':FP, 'FN':FN, 'TP':TP,},ignore_index=True)
              print()
              print()
```



```
In [155]: Score.sort_values("Recall Score",ascending=False)

Out[155]:
```

	Name	Train Score	Test Score	Precision Score	Recall Score	F1-Score	ROC - AUC	TN	FP	FN	TP
4	LGBMClassifier()	0.821480	0.783403	0.556059	0.528525	0.541943	0.787823	2511	392	438	491
5	GaussianNB()	0.720671	0.702505	0.405551	0.487621	0.442815	0.690309	2239	664	476	453
6	GradientBoostingClassifier()	0.796359	0.788100	0.575679	0.479010	0.522914	0.787289	2575	328	484	445
3	XGBClassifier()	0.861803	0.773225	0.537594	0.461787	0.496815	0.773295	2534	369	500	429
2	RandomForestClassifier()	0.986494	0.765136	0.518807	0.430571	0.470588	0.751686	2532	371	529	400
1	DecisionTreeClassifier()	0.986559	0.715814	0.415433	0.423036	0.419200	0.620038	2350	553	536	393
8	KNeighborsClassifier()	0.826373	0.741910	0.460000	0.371367	0.410959	0.701227	2498	405	584	345
7	AdaBoostClassifier()	0.776589	0.778445	0.586580	0.291712	0.389648	0.784991	2712	191	658	271
0	LogisticRegression()	0.761712	0.762787	0.525907	0.218515	0.308745	0.746843	2720	183	726	203

These are the model scores of the above defined model sorted on the basis of accuracy.

5.3 ROC AUC ANALYSIS

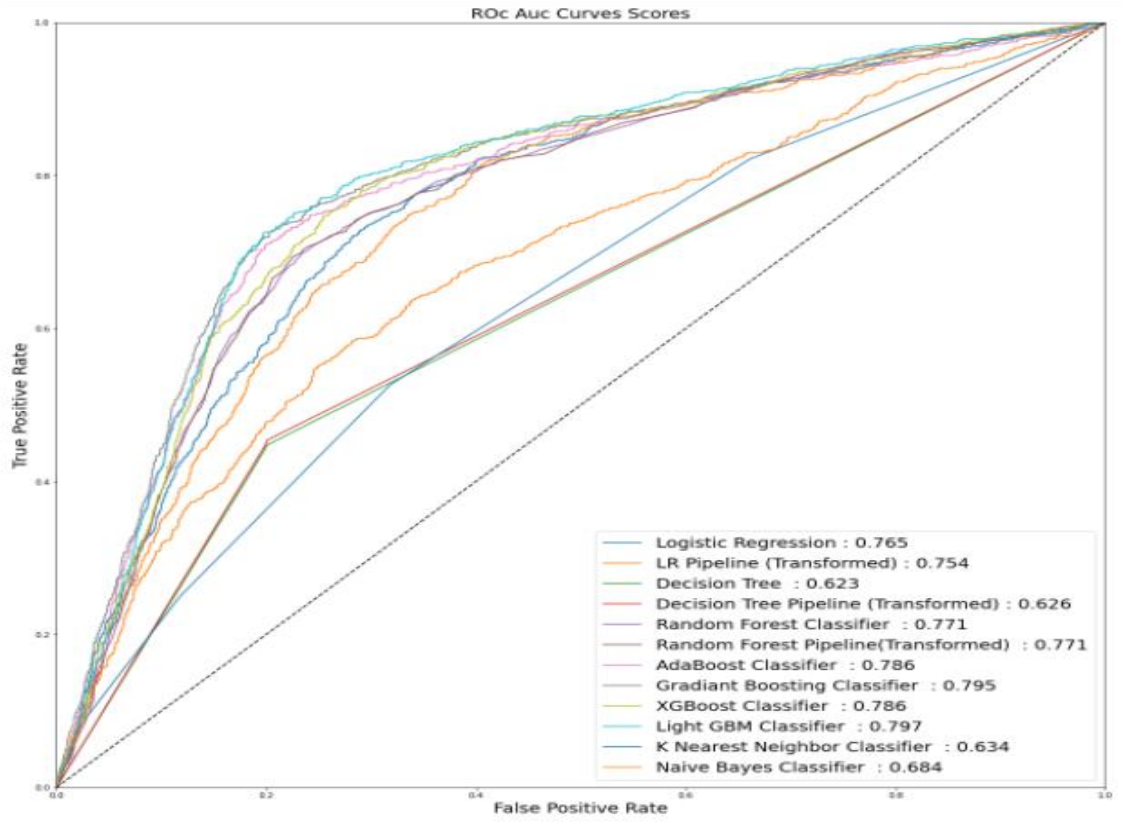
What is the AUC - ROC Curve?

AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separation. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. By analogy, the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.

The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.

How to speculate about the performance of the model?

An excellent model has AUC near to the 1 which means it has a good measure of separation. A poor model has an AUC near 0 which means it has the worst measure of separation. In fact, it means it is reciprocating the result. It is predicting 0s as 1s and 1s as 0s. And when AUC is 0.5, it means the model has no class separation capacity whatsoever.



From the above AUC analysis, we can see that the n KNN has the highest accuracy but doesn’t perform so well in the Auc analysis.

Here, we can see that the LightGBM algorithm is working best in terms of AUC , recall , accuracy. However, our data is imbalanced. Hence, we balance the data first and then again compare recall score, precision score and Auc to choose the algorithm for our final model

5.4 HANDLING DATA IMBALANCE:

Oversampling and **undersampling** in data analysis are techniques used to adjust the class distribution of a data set (i.e. the ratio between the different classes/categories represented). These terms are used both in statistical sampling, survey design methodology and in machine learning.

Oversampling and undersampling are opposite and roughly equivalent techniques.

Oversampling is generally employed more frequently than undersampling, especially when the detailed data has yet to be collected by survey, interview or otherwise. Undersampling is employed much less frequently. Overabundance of already collected data became an issue only in the "Big Data" era, and the reasons to use undersampling are mainly practical and related to resource costs. Specifically, while one needs a suitably large sample size to draw valid statistical conclusions, the data must be cleaned before it can be used. Cleansing typically involves a significant human component, and is typically specific to the dataset and the analytical problem, and therefore takes time and money.

Why over sampling??

Imbalanced datasets are those where there is a severe skew in the class distribution, such as 1:100 or 1:1000 examples in the minority class to the majority class.

This bias in the training dataset can influence many machine learning algorithms, leading some to ignore the minority class entirely. This is a problem as it is typically the minority class on which predictions are most important.

One approach to addressing the problem of class imbalance is to randomly resample the training dataset. The two main approaches to randomly resampling an imbalanced dataset are to delete examples from the majority class, called undersampling, and to duplicate examples from the minority class, called oversampling.

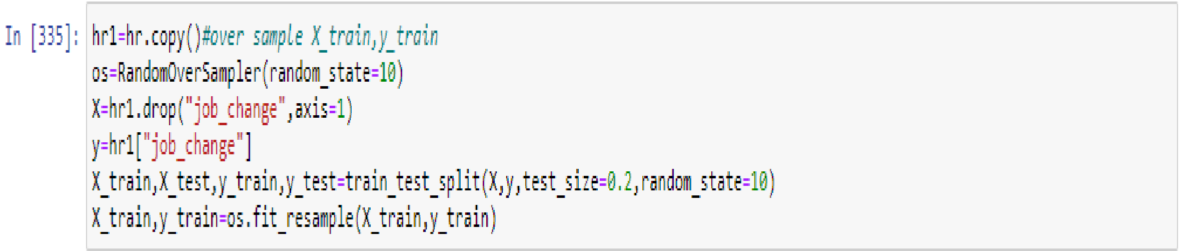
Random Oversampling Imbalanced Datasets

Random oversampling involves randomly duplicating examples from the minority class and adding them to the training dataset.

Examples from the training dataset are selected randomly with replacement. This means that examples from the minority class can be chosen and added to the new "*more balanced*" training dataset multiple times; they are selected from the original training dataset, added to the new training dataset, and then returned or "*replaced*" in the original dataset, allowing them to be selected again.

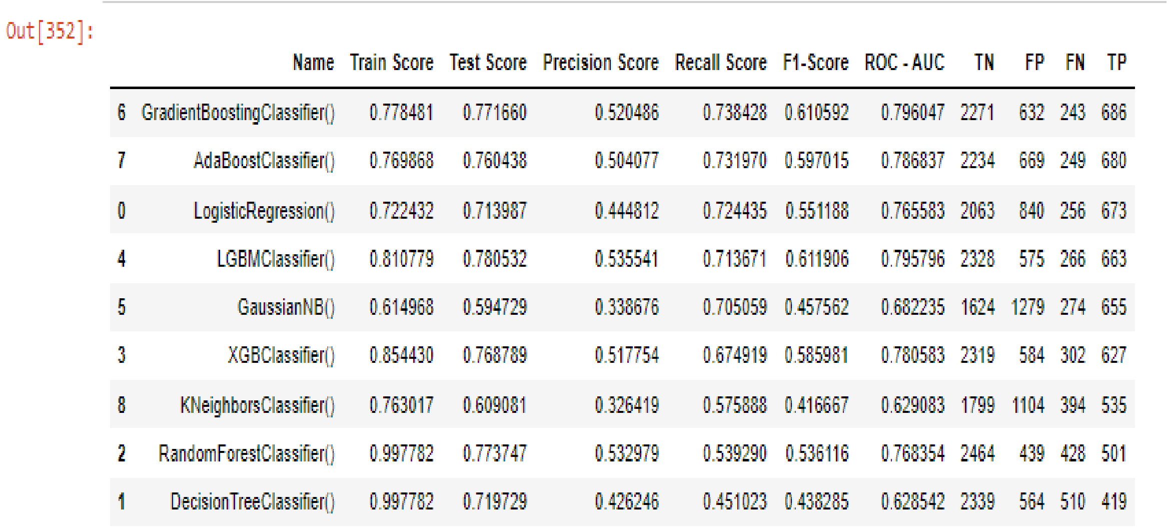
This technique can be effective for those machine learning algorithms that are affected by a skewed distribution and where multiple duplicate examples for a given class can influence the fit of the model. This might include algorithms that iteratively learn coefficients, like artificial neural networks that use stochastic gradient descent. It can also affect models that seek good splits of the data, such as support vector machines and decision trees.

It might be useful to tune the target class distribution. In some cases, seeking a balanced distribution for a severely imbalanced dataset can cause affected algorithms to overfit the minority class, leading to increased generalization error. The effect can be better performance on the training dataset, but worse performance on the holdout or test dataset.



```
In [335]: hr1=hr.copy()#over sample X_train,y_train
          os=RandomOverSampler(random_state=10)
          X=hr1.drop("job_change",axis=1)
          y=hr1["job_change"]
          X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=10)
          X_train,y_train=os.fit_resample(X_train,y_train)
```

Post oversampling report of model performance



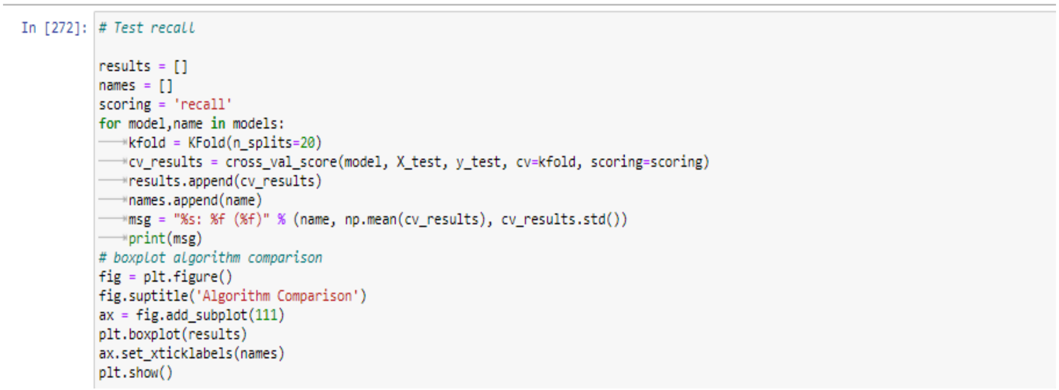
Out[352]:

	Name	Train Score	Test Score	Precision Score	Recall Score	F1-Score	ROC - AUC	TN	FP	FN	TP
6	GradientBoostingClassifier()	0.778481	0.771660	0.520486	0.738428	0.610592	0.796047	2271	632	243	686
7	AdaBoostClassifier()	0.769868	0.760438	0.504077	0.731970	0.597015	0.786837	2234	669	249	680
0	LogisticRegression()	0.722432	0.713987	0.444812	0.724435	0.551188	0.765583	2063	840	256	673
4	LGBMClassifier()	0.810779	0.780532	0.535541	0.713671	0.611906	0.795796	2328	575	266	663
5	GaussianNB()	0.614968	0.594729	0.338676	0.705059	0.457562	0.682235	1624	1279	274	655
3	XGBClassifier()	0.854430	0.768789	0.517754	0.674919	0.585981	0.780583	2319	584	302	627
8	KNeighborsClassifier()	0.763017	0.609081	0.326419	0.575888	0.416667	0.629083	1799	1104	394	535
2	RandomForestClassifier()	0.997782	0.773747	0.532979	0.539290	0.536116	0.768354	2464	439	428	501
1	DecisionTreeClassifier()	0.997782	0.719729	0.426246	0.451023	0.438285	0.628542	2339	564	510	419

Post over sampling we can clearly see the increase in the recall scores comapred to the past recall scores so the model are learning better and this is a step in the right direction the next step will be to compare all the models with each other to choose the opptimal final model .

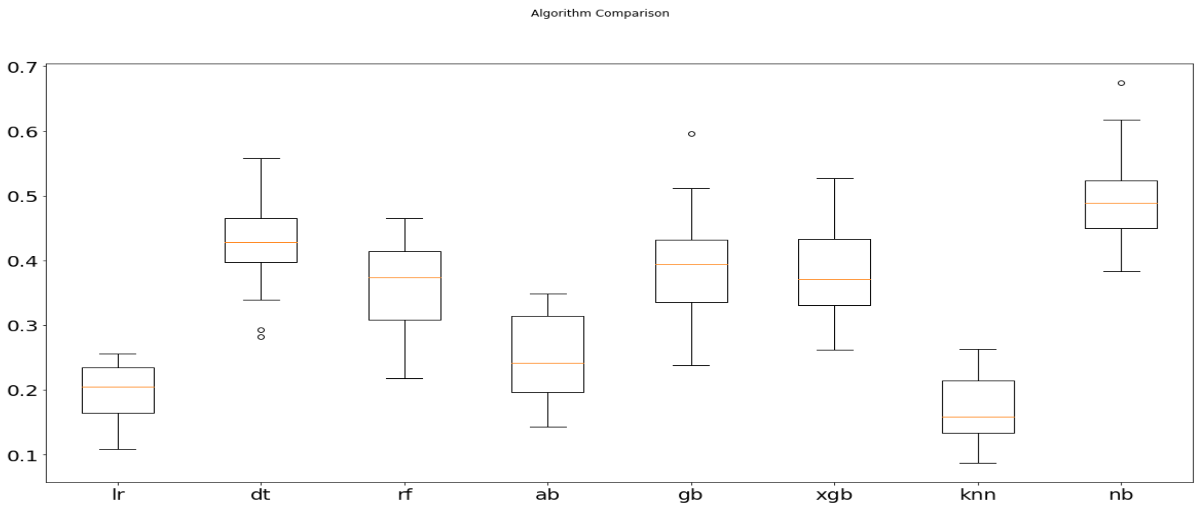
<u>Model Comparison Study based on the accuracy scores</u>

Recall Comparision After Oversampling



In [272]:

```
# Test recall

results = []
names = []
scoring = 'recall'
for model,name in models:
    kfold = KFold(n_splits=20)
    cv_results = cross_val_score(model, X_test, y_test, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, np.mean(cv_results), cv_results.std())
    print(msg)
# boxplot algorithm comparison
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```



Model Comparison Overall Analysis

Form the above model comaprision we can distinctly see that the rf models is giving the highest accuracy followed by the dt models which works similar to that and followed by the xgb model which is an boosting algorithm.as also we can see that gradient boosting is performing best on roc as well.

5.5 HYPER PARATMETER TUNING

What is hyper parameter tuning ??

In machine learning **hyper-parameter optimization** or tuning is the problem of choosing a set of optimal hyper-parameters for a learning algorithm. A hyper-parameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned. The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns. These measures are called hyper-parameters, and have to be tuned so that the model can optimally solve the machine learning problem. Hyper-parameter optimization finds a tuple of hyper-parameters that yields an optimal model which minimizes a predefined loss function on given independent data. The objective function takes a tuple of hyper-parameters and returns the associated loss. Cross-validation is often used to estimate this generalization performance.

What is the importance of hyper-parameter tuning?

Hyper-parameters are crucial as they control the overall behaviour of a machine learning model. The ultimate goal is to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results.Failure to do so would give sub-optimal results as the model didn't converge and minimize the loss function effectively.

We have selected Gradient Boosting Algorithm, its hyper parameters are as follows :

1. min_samples_split

- Defines the minimum number of samples (or observations) which are required in a node to be considered for splitting.
- Used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.
- Too high values can lead to under-fitting hence, it should be tuned using CV.

2. min_samples_leaf

- Defines the minimum samples (or observations) required in a terminal node or leaf.
- Used to control over-fitting similar to min_samples_split.
- Generally lower values should be chosen for imbalanced class problems because the regions in which the minority class will be in majority will be very small.

3.min_weight_fraction_leaf

- Similar to min_samples_leaf but defined as a fraction of the total number of observations instead of an integer.
- Only one of #2 and #3 should be defined.

4. max_depth

- The maximum depth of a tree.
- Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
- Should be tuned using CV.

6. max_leaf_nodes

- The maximum number of terminal nodes or leaves in a tree.
- Can be defined in place of max_depth. Since binary trees are created, a depth of ‘n’ would produce a maximum of 2^n leaves.
- If this is defined, GBM will ignore max_depth.

7. max_features

- The number of features to consider while searching for a best split. These will be randomly selected.
- As a thumb-rule, square root of the total number of features works great but we should check upto 30-40% of the total number of features.
- Higher values can lead to over-fitting but depends on case to case.

8. learning_rate

- This determines the impact of each tree on the final outcome . GBM works by starting with an initial estimate which is updated using the output of each tree. The learning parameter controls the magnitude of this change in the estimates.
- Lower values are generally preferred as they make the model robust to the specific characteristics of tree and thus allowing it to generalize well.
- Lower values would require higher number of trees to model all the relations and will be computationally expensive.

9. n_estimators

- The number of sequential trees to be modeled
- Though GBM is fairly robust at higher number of trees but it can still overfit at a point. Hence, this should be tuned using CV for a particular learning rate.

5.6 VALIDATION CURVES

What is Validation curve?

A **Validation Curve** is an important diagnostic tool that shows the sensitivity between to changes in a Machine Learning model’s accuracy with change in some parameter of the model.

A validation curve is typically drawn between some parameter of the model and the model’s score. Two curves are present in a validation curve – one for the training set score and one for the cross-validation score. By default, the function for validation curve, present in the scikit-learn library performs 3-fold cross-validation.

A validation curve is used to evaluate an existing model based on hyper-parameters and is not used to tune a model. This is because, if we tune the model according to the validation score, the model may be biased towards the specific data against which the model is tuned; thereby, not being a good estimate of the generalization of the model.

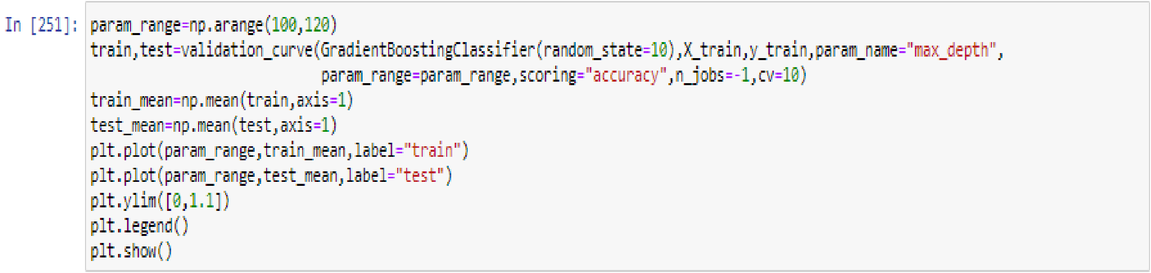
Interpreting a Validation Curve

Interpreting the results of a validation curve can sometimes be tricky. Keep the following points in mind while looking at a validation curve :

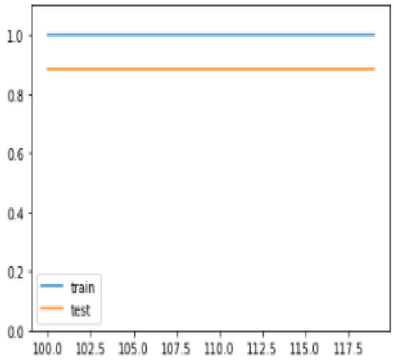
- Ideally, we would want both the validation curve and the training curve to look as similar as possible.
- If both scores are low, the model is likely to be ***underfitting***. This means either the model is too simple or it is informed by too few features. It could also be the case that the model is regularized too much.
- If the training curve reaches a high score relatively quickly and the validation curve is lagging behind, the model is ***overfitting.*** This means the model is very complex and there is too little data; or it could simply mean there is too little data.
- We would want the value of the parameter where the training and validation curves are closest to each other.

Gradient Boosting Classifier models validation curves

Max_depth Validation Curve

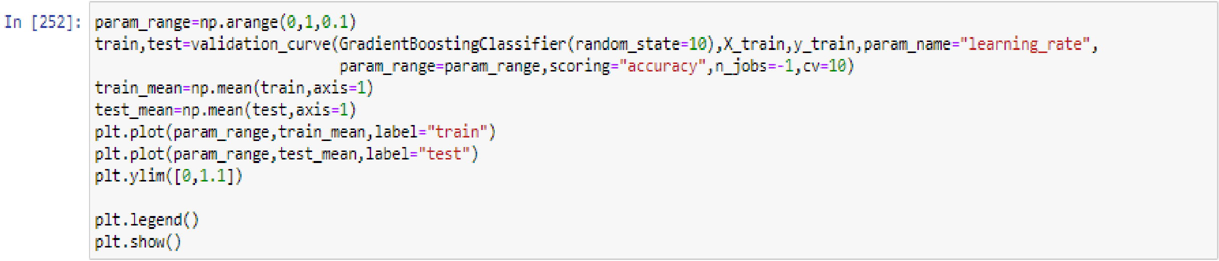


```
In [251]: param_range=np.arange(100,120)
          train,test=validation_curve(GradientBoostingClassifier(random_state=10),X_train,y_train,param_name="max_depth",
                                      param_range=param_range,scoring="accuracy",n_jobs=-1,cv=10)
          train_mean=np.mean(train,axis=1)
          test_mean=np.mean(test,axis=1)
          plt.plot(param_range,train_mean,label="train")
          plt.plot(param_range,test_mean,label="test")
          plt.ylim([0,1.1])
          plt.legend()
          plt.show()
```

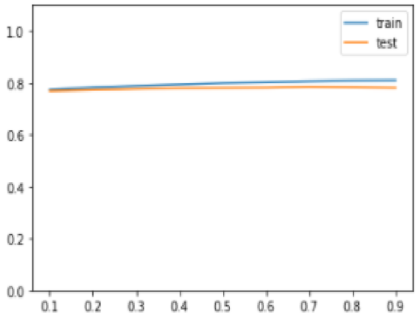


We can see the there is no change in the max depth over incereseing the number of max depth and it stays constant so we can choose any range.

Learning Rate Validation Curve

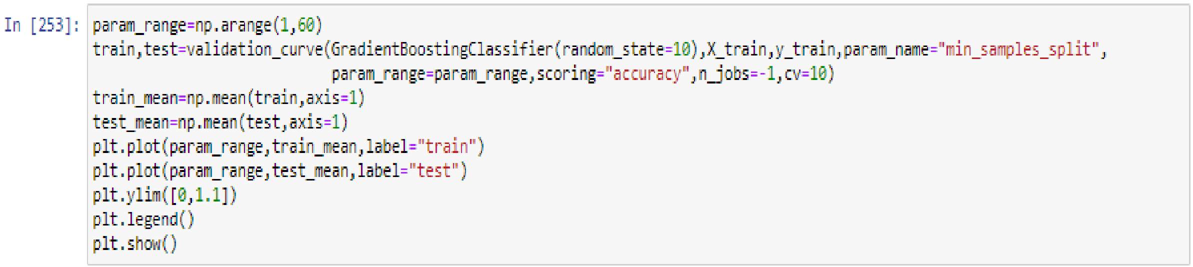


```
In [252]: param_range=np.arange(0,1,0.1)
          train,test=validation_curve(GradientBoostingClassifier(random_state=10),X_train,y_train,param_name="learning_rate",
                                      param_range=param_range,scoring="accuracy",n_jobs=-1,cv=10)
          train_mean=np.mean(train,axis=1)
          test_mean=np.mean(test,axis=1)
          plt.plot(param_range,train_mean,label="train")
          plt.plot(param_range,test_mean,label="test")
          plt.ylim([0,1.1])

          plt.legend()
          plt.show()
```

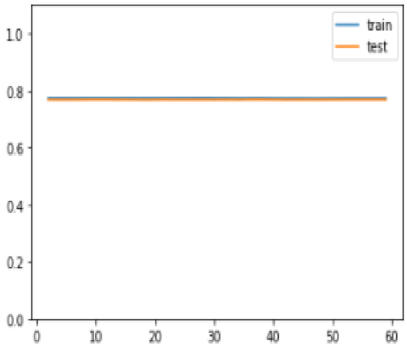


There is a increase in score when the learing rate incerases so we will keep it between 0 and 1

Min_samples split Validation curve

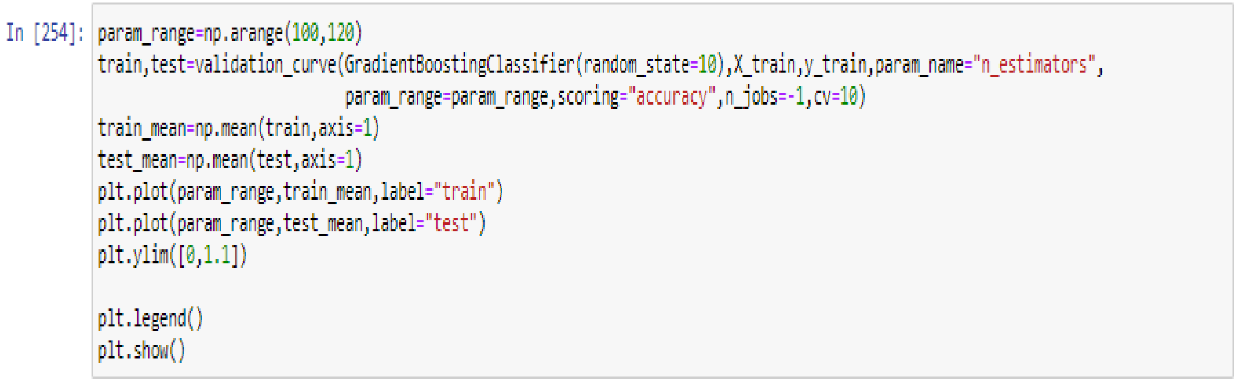


```
In [253]: param_range=np.arange(1,60)
          train,test=validation_curve(GradientBoostingClassifier(random_state=10),X_train,y_train,param_name="min_samples_split",
                                       param_range=param_range,scoring="accuracy",n_jobs=-1,cv=10)
          train_mean=np.mean(train,axis=1)
          test_mean=np.mean(test,axis=1)
          plt.plot(param_range,train_mean,label="train")
          plt.plot(param_range,test_mean,label="test")
          plt.ylim([0,1.1])
          plt.legend()
          plt.show()
```

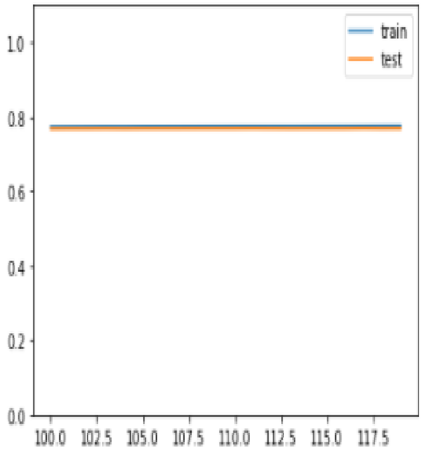


Both train and test scores are constant at around 80 so we can keep the min saples split 2-5

N_estimators validation Curve

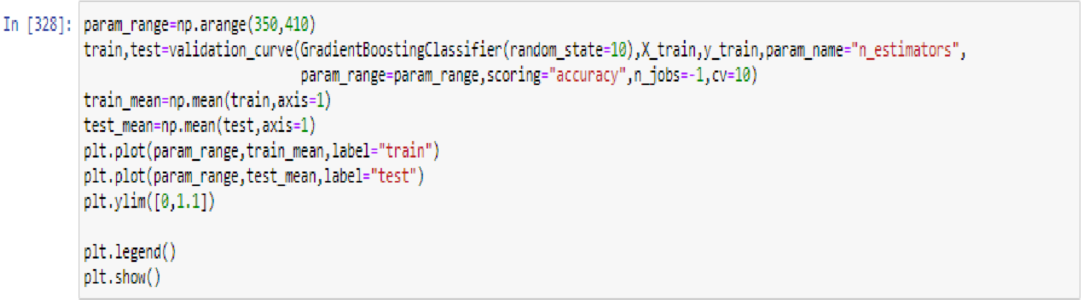


```
In [254]: param_range=np.arange(100,120)
          train,test=validation_curve(GradientBoostingClassifier(random_state=10),X_train,y_train,param_name="n_estimators",
                                       param_range=param_range,scoring="accuracy",n_jobs=-1,cv=10)
          train_mean=np.mean(train,axis=1)
          test_mean=np.mean(test,axis=1)
          plt.plot(param_range,train_mean,label="train")
          plt.plot(param_range,test_mean,label="test")
          plt.ylim([0,1.1])

          plt.legend()
          plt.show()
```



As the numer of estimatoes increse the scores increases so we will go for higher n_Estimators as here we can se that there is not much affect of n_Estimators on the model performance

Gradient Boosting Classifier models Grid Search CV

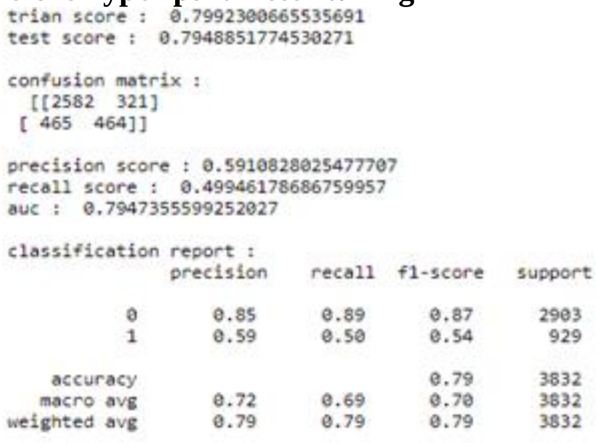


```
In [328]: param_range=np.arange(350,410)
          train,test=validation_curve(GradientBoostingClassifier(random_state=10),X_train,y_train,param_name="n_estimators",
                                      param_range=param_range,scoring="accuracy",n_jobs=-1,cv=10)
          train_mean=np.mean(train,axis=1)
          test_mean=np.mean(test,axis=1)
          plt.plot(param_range,train_mean,label="train")
          plt.plot(param_range,test_mean,label="test")
          plt.ylim([0,1.1])

          plt.legend()
          plt.show()
```

We choose to work on n-estimators, learning rate, min samples split and max depth and after that we have arrived at a end result that the best parameters for the learning rate is 0.13 min samples split of 3 and n-estimators of 400 while the max_depth is kept none so that the model properly understands the data as we are not getting a accuracy that might show us that the model is over-fitting so we have left the max_depth to None.

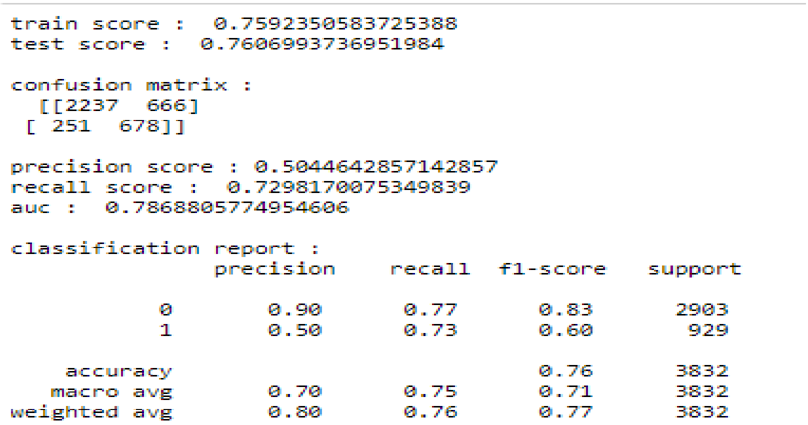
5.7 PRE AND POST METRICS OF HYPER PARAMETER TUNING:

Before hyper parameter tuning



```
trian score :  0.7992300665535691
test score :  0.7948851774530271

confusion matrix :
  [[2582  321]
 [ 465  464]]

precision score : 0.5910828025477707
recall score :  0.49946178686759957
auc :  0.7947355599252027

classification report :
               precision    recall  f1-score   support

           0       0.85      0.89      0.87      2903
           1       0.59      0.50      0.54       929

    accuracy                           0.79      3832
   macro avg       0.72      0.69      0.70      3832
weighted avg       0.79      0.79      0.79      3832
```

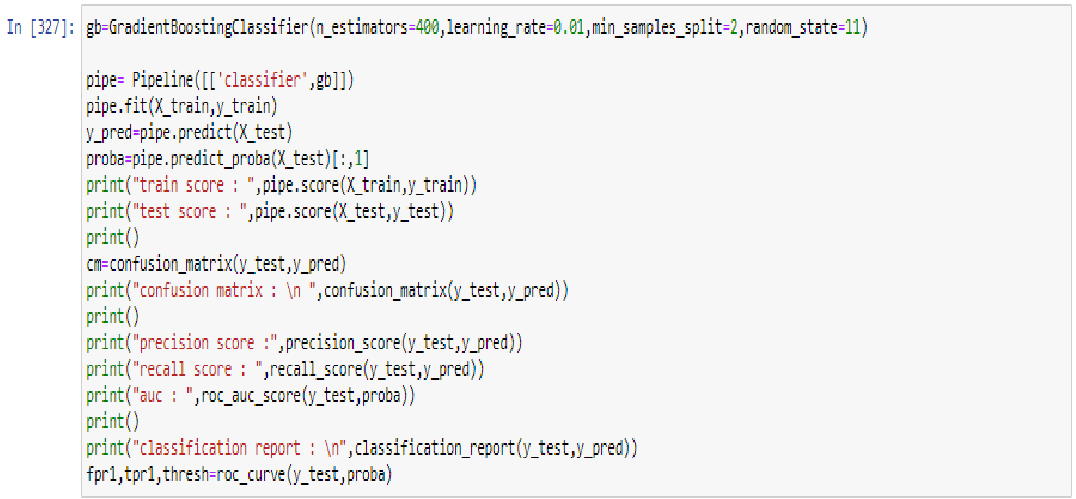
After hyper parameter tuning



```
train score :  0.7592350583725388
test score :  0.7606993736951984

confusion matrix :
  [[2237  666]
 [ 251  678]]

precision score : 0.5044642857142857
recall score :  0.7298170075349839
auc :  0.7868805774954606

classification report :
               precision    recall  f1-score   support

           0       0.90      0.77      0.83      2903
           1       0.50      0.73      0.60       929

    accuracy                           0.76      3832
   macro avg       0.70      0.75      0.71      3832
weighted avg       0.80      0.76      0.77      3832
```

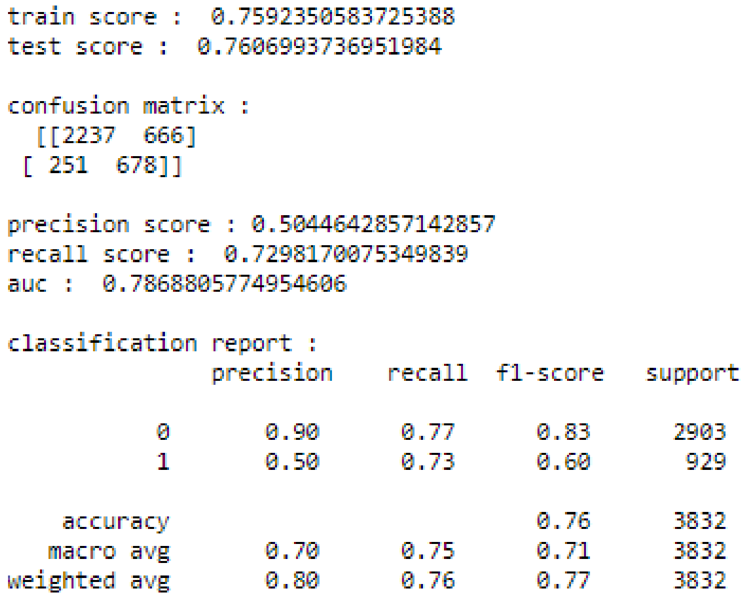
We can clearly observe the difference in the two, after oversampling and hyperparameter tuning not only the overfitting of the data and there is difference in test and train score as well as we can see that there has beena significant increase in the recall score after performing the following steps

6. FINAL MODEL FIT



```
In [327]: gb=GradientBoostingClassifier(n_estimators=400,learning_rate=0.01,min_samples_split=2,random_state=11)

          pipe= Pipeline([['classifier',gb]])
          pipe.fit(X_train,y_train)
          y_pred=pipe.predict(X_test)
          proba=pipe.predict_proba(X_test)[:,1]
          print("train score : ",pipe.score(X_train,y_train))
          print("test score : ",pipe.score(X_test,y_test))
          print()
          cm=confusion_matrix(y_test,y_pred)
          print("confusion matrix : \n ",confusion_matrix(y_test,y_pred))
          print()
          print("precision score :",precision_score(y_test,y_pred))
          print("recall score : ",recall_score(y_test,y_pred))
          print("auc : ",roc_auc_score(y_test,proba))
          print()
          print("classification report : \n",classification_report(y_test,y_pred))
          fpr1,tpr1,thresh=roc_curve(y_test,proba)
```

Reuslt from the final model

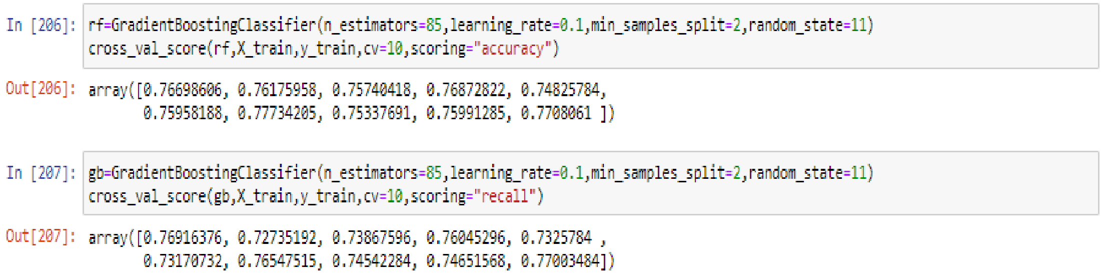


```
train score :  0.7592350583725388
test score :  0.7606993736951984

confusion matrix :
   [[2237  666]
 [ 251  678]]

precision score : 0.5044642857142857
recall score :  0.7298170075349839
auc :  0.7868805774954606

classification report :
               precision    recall  f1-score   support

           0       0.90      0.77      0.83      2903
           1       0.50      0.73      0.60       929

    accuracy                           0.76      3832
   macro avg       0.70      0.75      0.71      3832
weighted avg       0.80      0.76      0.77      3832
```

Here we can see that the model is not overfit not underfit while the model is perfectly fitting out data and giving a desireable reacall and AUC score as well which is the best metrics which are important for the final model th perform good in real life scenarios.

Crossvalidation of the scores to check that the model is consistent and is a generalized model .

Final Cross Validation Score



```
In [206]: rf=GradientBoostingClassifier(n_estimators=85,learning_rate=0.1,min_samples_split=2,random_state=11)
          cross_val_score(rf,X_train,y_train,cv=10,scoring="accuracy")

Out[206]: array([0.76698606, 0.76175958, 0.75740418, 0.76872822, 0.74825784,
                 0.75958188, 0.77734205, 0.75337691, 0.75991285, 0.7708061 ])

In [207]: gb=GradientBoostingClassifier(n_estimators=85,learning_rate=0.1,min_samples_split=2,random_state=11)
          cross_val_score(gb,X_train,y_train,cv=10,scoring="recall")

Out[207]: array([0.76916376, 0.72735192, 0.73867596, 0.76045296, 0.7325784 ,
                 0.73170732, 0.76547515, 0.74542284, 0.74651568, 0.77003484])
```

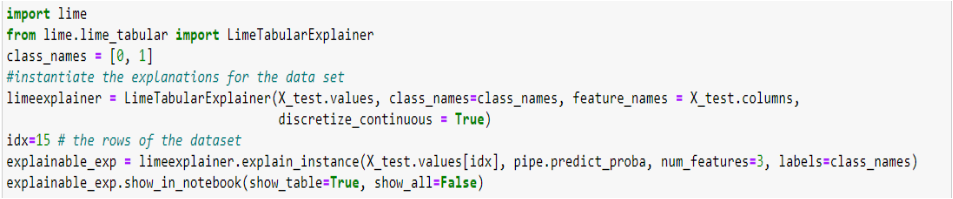
7. MODEL INTERPRETIBILITY

In a sense, every time an engineer loads a machine learning model in production, the engineer fully trusts that the model will make reasonable predictions. Such tests are usually performed by looking at the expected accuracy or other method of integration. However, as anyone who has ever used machine learning in a real app can attest, such metrics can be very misleading. Sometimes data that should not be accidentally obtained is leaked to training and specific information (e.g., future look). Sometimes a model makes mistakes that are too embarrassing to accept. These and many other tricky problems show that understanding model prediction can be another useful tool when deciding whether a model is reliable or not, because people tend to have good intuition and business acumen that are hard to take on test metrics.

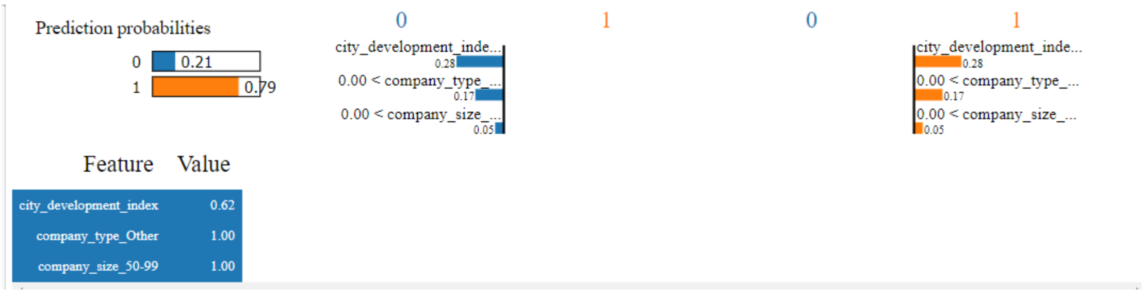
7.1 LIMEw

To overcome such issue, We propose Local Interpretable Model-Agnostic Explanations (LIME), a technique to explain the predictions of any machine learning classifier, and evaluate its usefulness in various tasks related to trust.

Intuitively, an explanation is a local linear approximation of the model's behaviour. While the model may be very complex globally, it is easier to approximate it around the vicinity of a particular instance. While treating the model as a black box, LIME perturbs the instance desired to explain and learn a sparse linear model around it, as an explanation. We then learn a linear model (dashed line) that approximates the model well in the vicinity of our data.



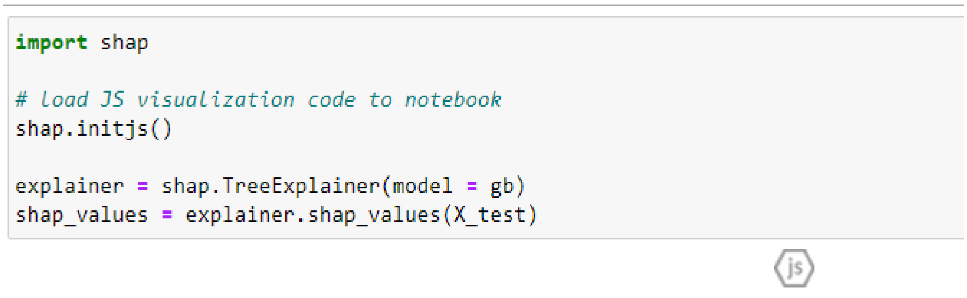
```
import lime
from lime.lime_tabular import LimeTabularExplainer
class_names = [0, 1]
#instantiate the explanations for the data set
limeexplainer = LimeTabularExplainer(X_test.values, class_names=class_names, feature_names = X_test.columns,
                                     discretize_continuous = True)
idx=15 # the rows of the dataset
explainable_exp = limeexplainer.explain_instance(X_test.values[idx], pipe.predict_proba, num_features=3, labels=class_names)
explainable_exp.show_in_notebook(show_table=True, show_all=False)
```



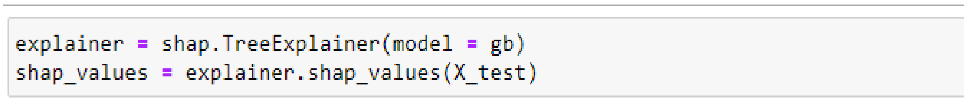
Feature	Value
city_development_index	0.62
company_type_Other	1.00
company_size_50-99	1.00

7.2 SHAP

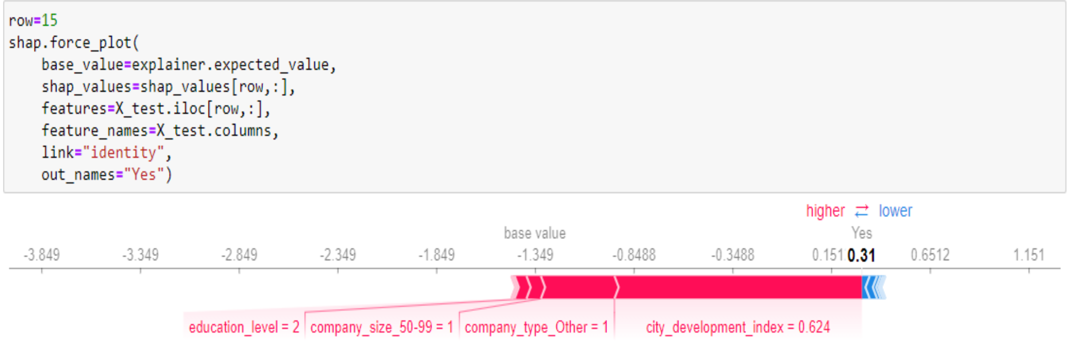
The effectiveness of SHAPley (SHapley Additive explanations) depends on the inclusion of all available frameworks for interpreting forecasts. SHAP gives each element the value of a specific forecast. Its novel elements include: The new category includes six existing modes, which are notable because a few of the most recent modes in the class do not have the proposed structures. Based on the insights from this integration, it introduces new ways of demonstrating improved computer performance and or better alignment with human emotion than previous methods.



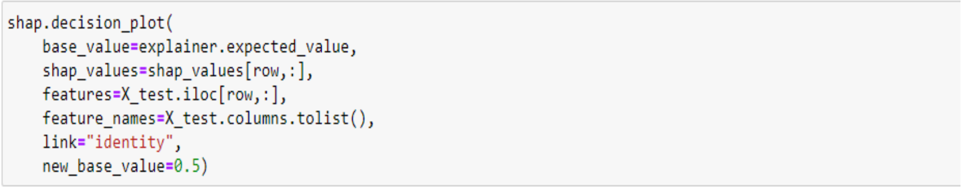
```
import shap

# load JS visualization code to notebook
shap.initjs()

explainer = shap.TreeExplainer(model = gb)
shap_values = explainer.shap_values(X_test)
```



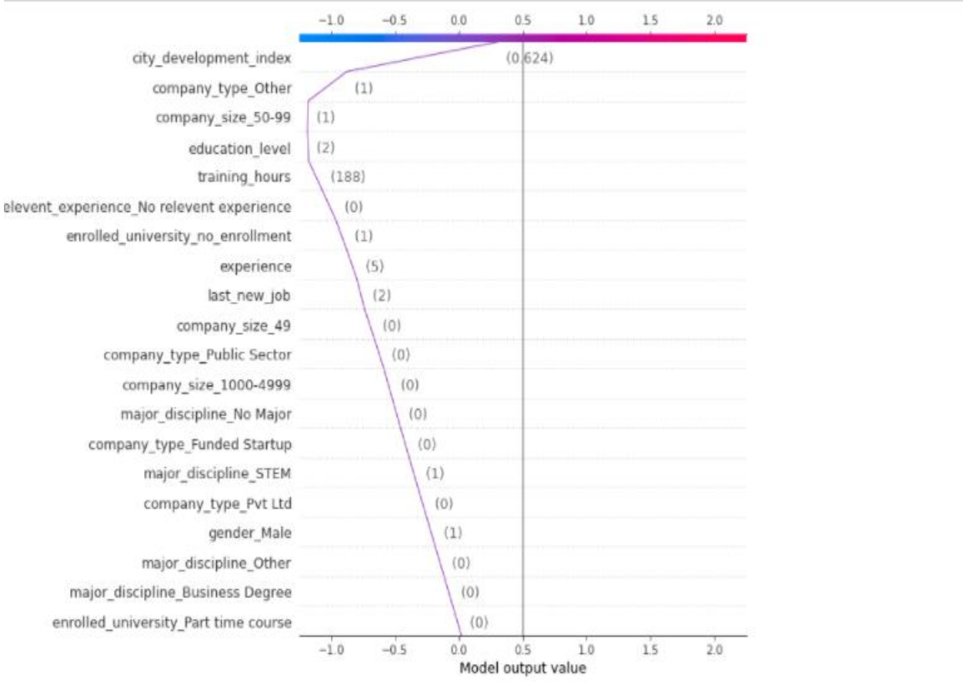
```
explainer = shap.TreeExplainer(model = gb)
shap_values = explainer.shap_values(X_test)
```



```
row=15
shap.force_plot(
    base_value=explainer.expected_value,
    shap_values=shap_values[row,:],
    features=X_test.iloc[row,:],
    feature_names=X_test.columns,
    link="identity",
    out_names="Yes")
```



```
shap.decision_plot(
    base_value=explainer.expected_value,
    shap_values=shap_values[row,:],
    features=X_test.iloc[row,:],
    feature_names=X_test.columns.tolist(),
    link="identity",
    new_base_value=0.5)
```



Conclusion

Employee’s job switch has been identified as a pivotal factor to curb the growth of organizations. In this research, the performance of seven supervised machine learning models are evaluated on the HR dataset. In addition to statistical analysis, a number of data mining techniques are introduced and used in this study, including data scaling, parameter searching and cross validation. To enhance the interpretability of employee’s job switch model. Extreme Gradient Boosting is recommended to use as the most reliable algorithm and it is our final model. It requires minimal data preprocessing and also have decent predictive power, and ranks the feature importance automatically and reliably and ultimately the model needs to be used comes down to the client and their expectations and business problem.

Implications :

Our solution tells whether the company’s employee will switch the job or not. Our model Xgboost helped in the fast development and high performance which helps so to save the time of the organisation in the prediction. It solves the complex non linear functions and provides a better solution and high accuracy which will reduce the cost of the organisation and making the organisation aware about the employee’s the willing for job change and not willing for a job change.

Limitations :

The model requires some amount of hyper parameter tuning and makes it complex which can be difficult to understand and requires a good explanation. In our model we have to manually create dummy variable/ label encoding for categorical features before feeding them into the models which makes the model more complex as we do not know the actual values or names, hence while providing inference on the data we have to provide inference based on the encoded values.