Lab 1 Report

Name: Jay Bharadva

Student Id: 015958449

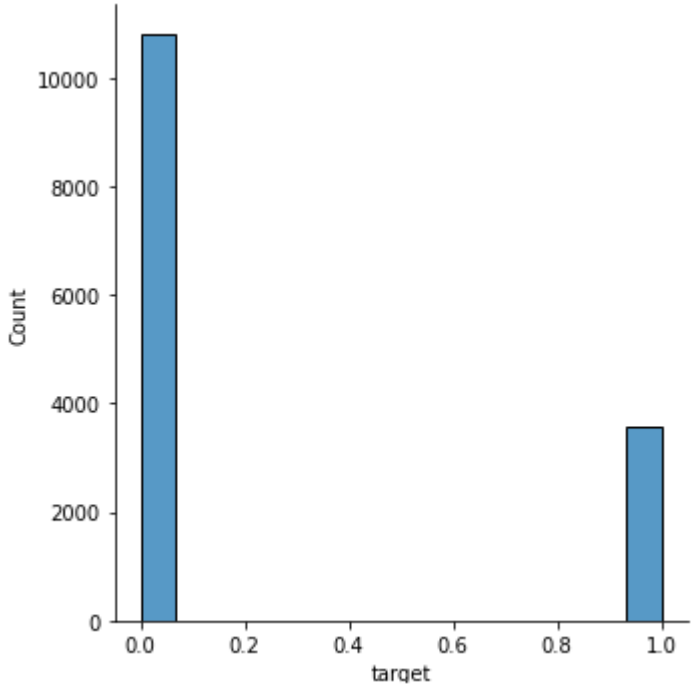
Dataset Summary and Observations:

Training Data Size: There are 14 columns and 14368 rows in Training data
Test Data Size: There are 13 columns and 4790 rows in Test Data

Summary of Training Data:

```
---------------------------------------------------------------------------------------------------
Min, max, avg, std dev etc. stats for continuous features...
       city_development_index  training_hours        target
count            14368.000000    14368.000000  14368.000000
mean                 0.828252       65.396645      0.247982
std                  0.123419       60.277583      0.431856
min                  0.448000        1.000000      0.000000
25%                  0.738000       23.000000      0.000000
50%                  0.899000       47.000000      0.000000
75%                  0.920000       88.000000      0.000000
max                  0.949000      336.000000      1.000000
---------------------------------------------------------------------------------------------------
```

Target Variable Distribution:

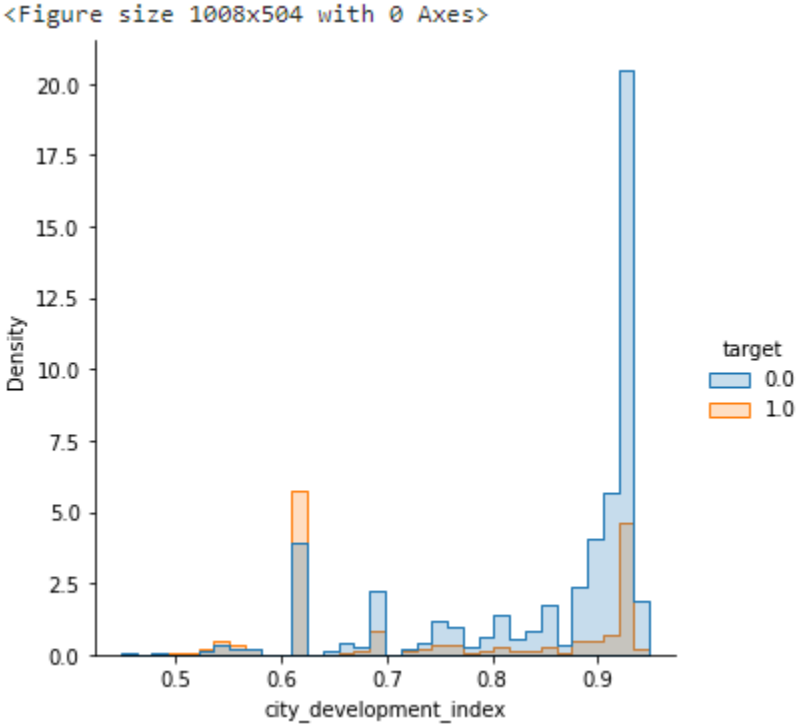


Target: 0 – Not looking for job change, 1 – Looking for a job change
Here, the target variable distribution is imbalanced.
Other variables with unfair distribution in training dataset:

City	Relevant_experience	Enrolled_university	Education_level
gender	Major_discipline	Company_type	

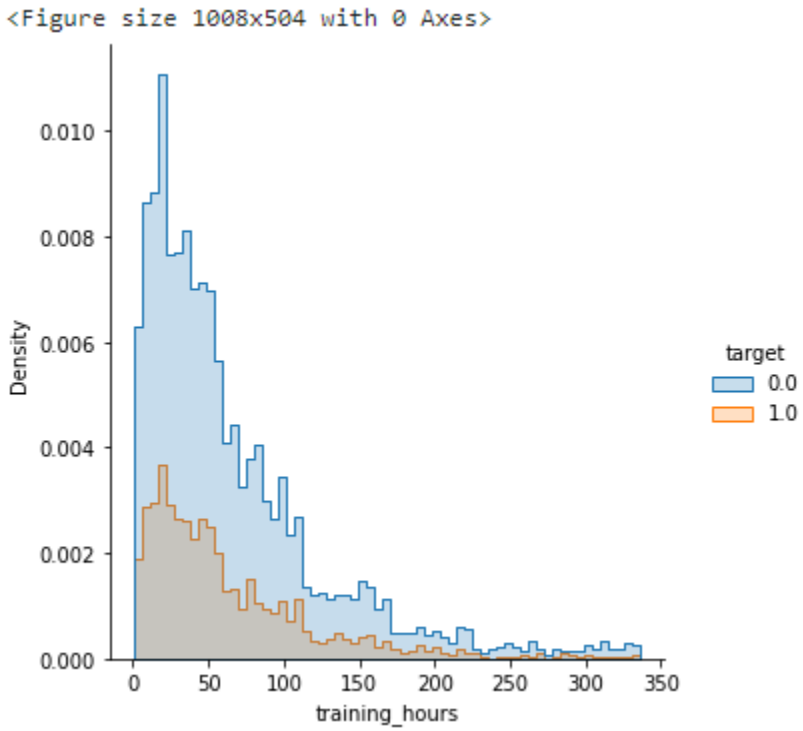
Data Visualization:

- **City_development_index vs target:**



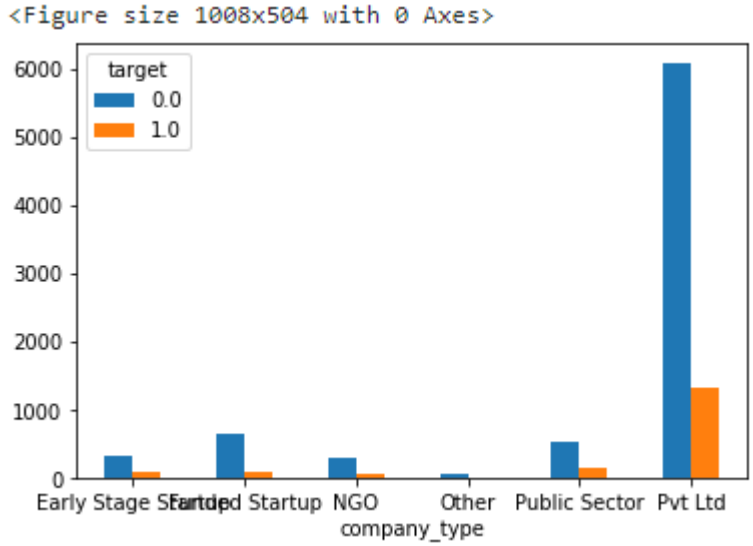
It is evident that most of the people are not looking for the job change where the development index of the city is higher

- **Training_hours vs target:**



Person who has invested much time in the training are not looking for job and people with less training hours are looking for a job change.

- **Company_size vs target:**



It is a clear observation that majority of the people are working in the private limited companies and most of them are not looking for a job change.

Preprocessing the dataset variables:

- **Experience:** replacing >20 with 21 and <1 with 0.5.
- **last_new_job:** replacing >4 with 5 and never with 0.
- **Company_size:** removing the special characters and converting the range to numeric**.** Taking a difference of upper number and lower number.
- **City:** replacing "City_" with "" and converting it to numbers.

Missing Values in the dataset:

	Null Values	Percentage of Missing Values
gender	3393	23.614978
enrolled_university	292	2.032294
education_level	338	2.352450
major_discipline	2089	14.539254
experience	45	0.313196
company_size	4430	30.832405
company_type	4598	32.001670
last_new_job	327	2.275891

- I have changed replaced all the null values with a corresponding mode of the data.

Transforming the categorical variable:

- **Dummies:** gender, enrolled_university, major_discipline
- **Mapping with meaningful order:** relevent_experience, company_size, education_level

Scaling the data:

- For LogisticRegression and Liner SVC model I have used MinMacScalar to scale the data

Classification Models Used in the Lab:

- Below table describes every model used in the lab with their corresponding best accuracy score obtained from GridSearchCV.

Classification Model	Best Score Obtained from Hyperparameter tuning
1. Support Vector Machine (SVC)	0.7811050189793336
2. Logistic Regression	0.7701391817798398
3. GaussianNB	0.7570645297342893
4. DecisionTreeClassifier	0.7933361450864614
5. RandomForestClassifier	0.7823631628041268
6. LogisticRegression(Implemented)	0.7501054407423028

Logistic Regression Implementation:

```
class LogisticRegression:
    def sigmoid(self,z):
        s = 1/(1+exp(-z))
        return s
    def initialize(self,X):
        w = np.zeros((X.shape[1]+1,1))
        X = np.c_[np.ones((X.shape[0],1)),X]
        return w,X
    def fit(self,X,y,alpha=0.01,iter=200):
        w,X = self.initialize(X)
        def cost(theta):
            z = dot(X,theta)
            cost0 = y.T.dot(log(self.sigmoid(z)))
            cost1 = (1-y).T.dot(log(1-self.sigmoid(z)))
            cost = -((cost1 + cost0))/len(y)
            return cost
        costL = np.zeros(iter,)
        for i in range(iter):
            w = w - alpha*dot(X.T,self.sigmoid(dot(X,w))-np.reshape(y,(len(y),1)))
            costL[i] = cost(w)
        self.w = w
        return costL
    def predict(self,X):
        z = dot(self.initialize(X)[1],self.w)
        y = []
        for i in self.sigmoid(z):
            if i>0.5:
                y.append(1)
            else:
                y.append(0)
        return y
```

Model Used for Final Submission:

- I used hyper parameter tuned Random Forest Classifier model for the final submission of the prediction of target variable.
- Parameter Dictionary used: {'n_estimators': [200, 500], 'max_features': ['auto', 'sqrt', 'log2'], 'max_depth' : [4,5,6,7,8], 'criterion' :['gini', 'entropy']}
- Accuracy on 60% of the test data: 0.51586

Overall Experience of Lab1:

It was very helpful that the dataset given to us was similar to that of the real-world data with unfair distribution and missing values. It was a good practice to prepare the dataset for classification model. Got a chance to get familiarize with the Kaggle competition environment.

Python Notebook Link:

https://github.com/0xLighty/CMPE257_Lab1