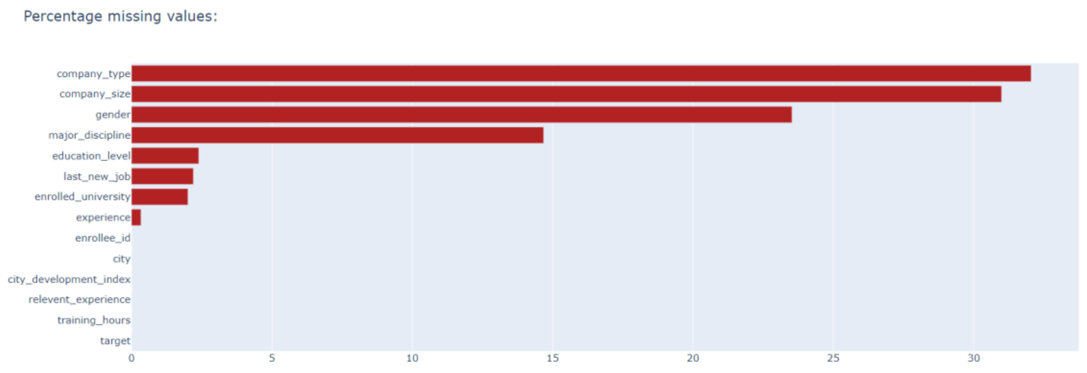
All visualizations:

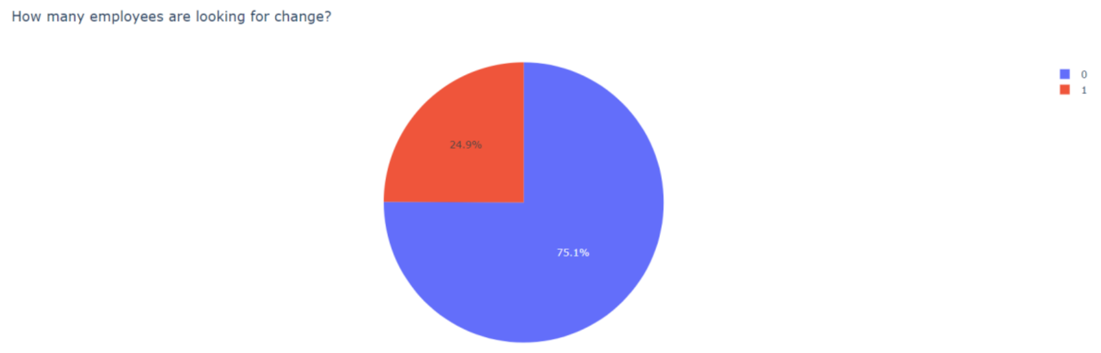
```
[240] missing = pd.concat([df_train.isnull().sum(), 100 * df_train.isnull().sum() / len(df_train)],
                axis=1).rename(columns={0:'Missing records', 1:'Percentage(%)'}).sort_values(by='Percentage(%)',
                                                                                          ascending=False)


      trace = go.Bar(y=missing.index[::-1], x=missing['Percentage(%)'][::-1],
                     orientation='h', marker=dict(color='firebrick',))
      data = [trace]
      layout = dict(title = 'Percentage missing values:', margin = dict(l = 200))

      fig = go.Figure(data = data, layout = layout)
      py.iplot(fig)
```



Visualizations

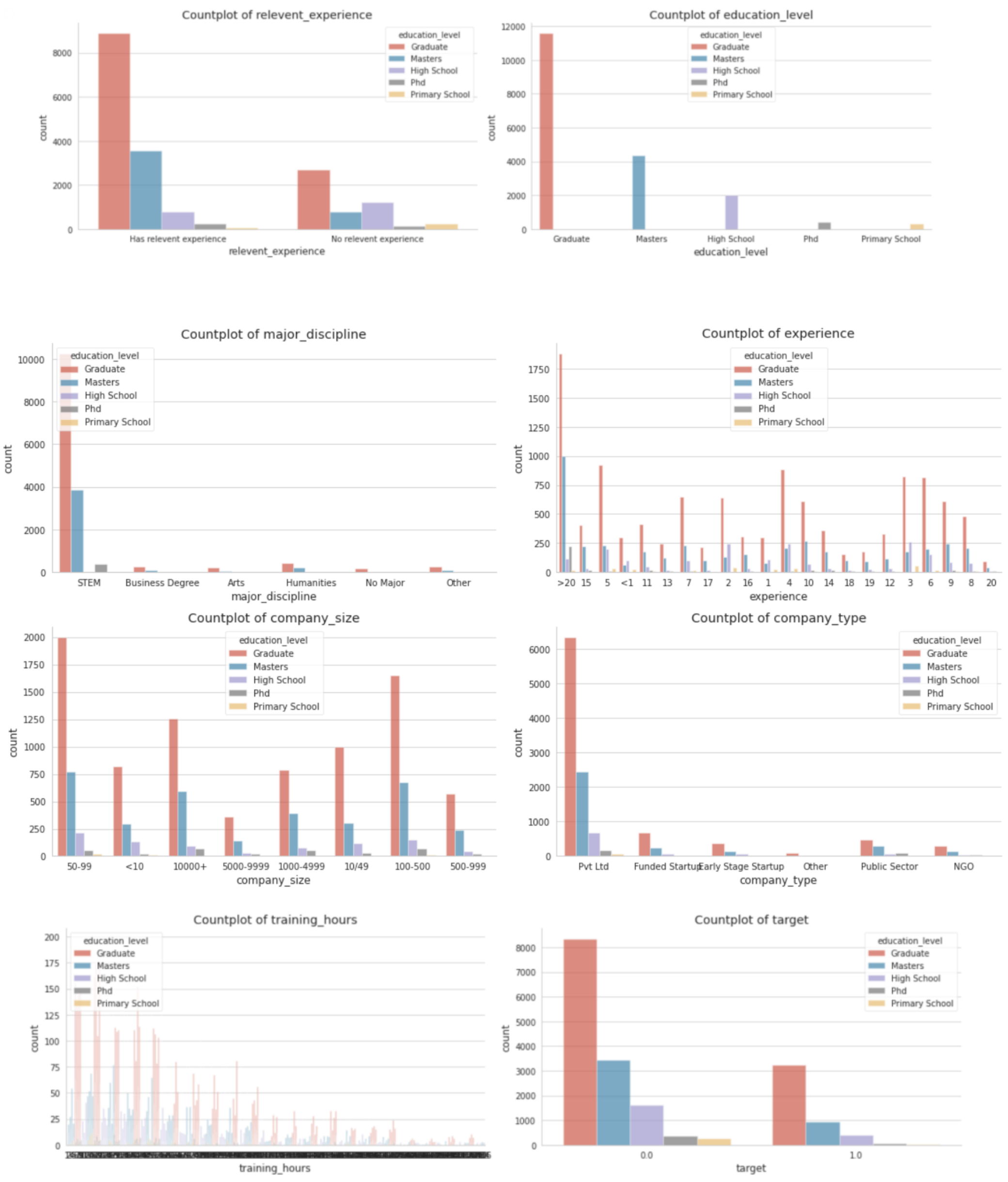
1. Piechart: how many employees are looking for change?

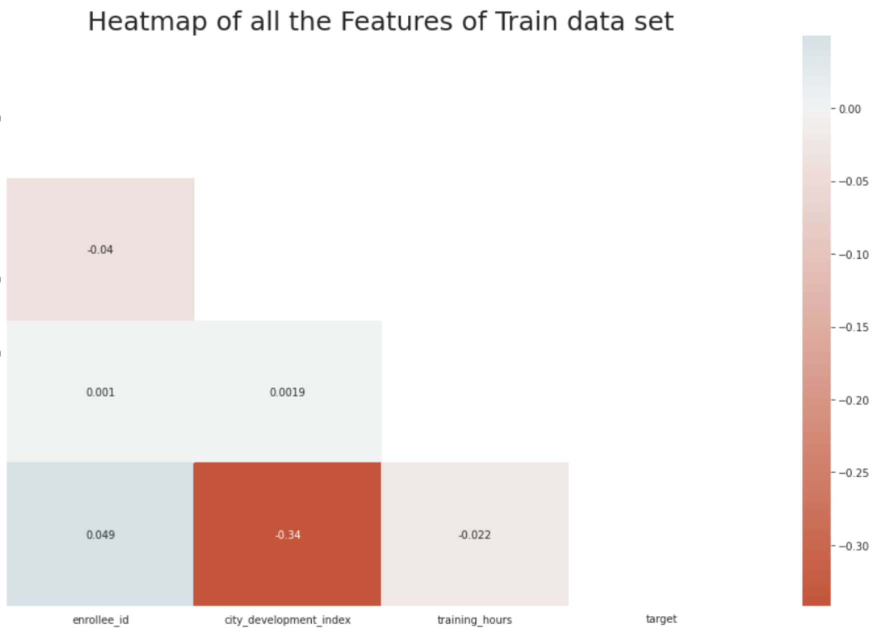
```
[242] tmp = df_train['target'].value_counts()

      trace = go.Pie(labels=list(tmp.index), values=list(tmp.values))
      layout = dict(title="How many employees are looking for change?")
      fig = dict(data=[trace], layout=layout)

      py.iplot(fig, filename="pie chart")
```



2. Countplots based on Educational level

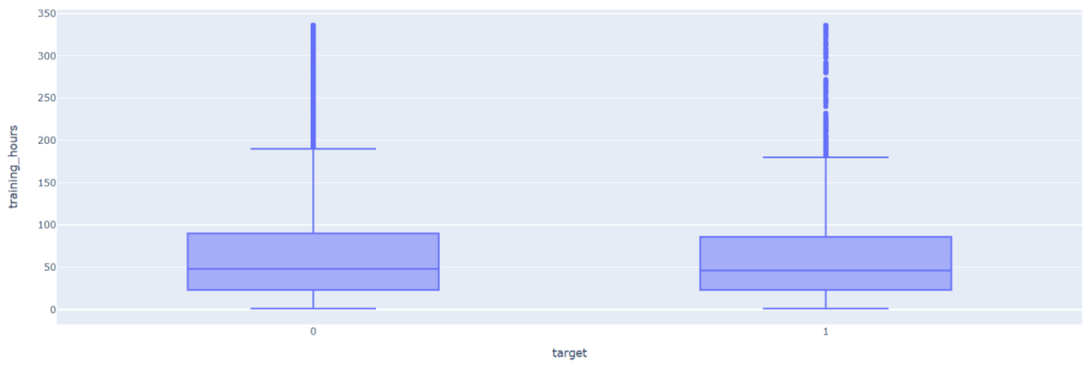
```
[243] plt.figure(figsize=[16,18])
      plot=["relevent_experience", "education_level","major_discipline", "experience","company_size","company_type", "training_hours","target"]

      n=1
      for f in plot:
        plt.subplot(4, 2, n)
        sns.countplot(x=f, hue='education_level', alpha=0.7, data=df_train)
        sns.despine()
        plt.title("Countplot of {}".format(f))
        n=n+1

      plt.tight_layout()
      plt.show()
```





4. Box plot for training hours of employees:

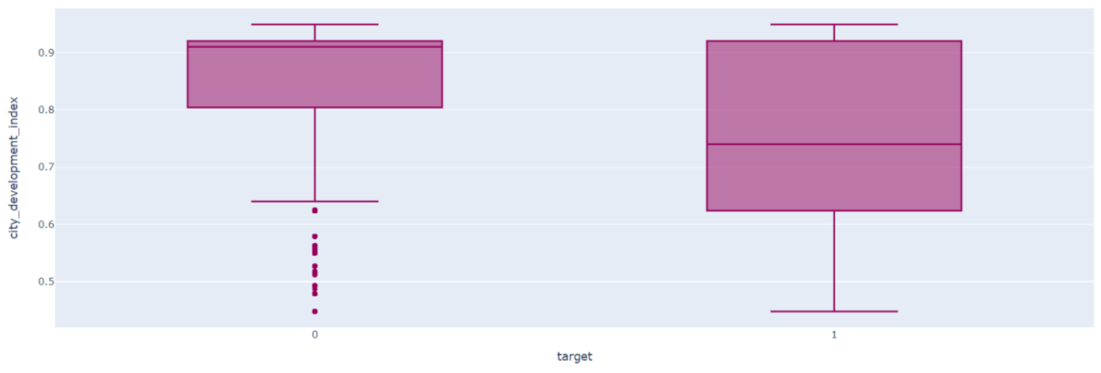
```
[246] px.box(data_frame=df_train, x='target', y='training_hours')
```



The number of training hours is same for both types of employees. thus it cannot help for prediction.

5. Box plot for city of employee

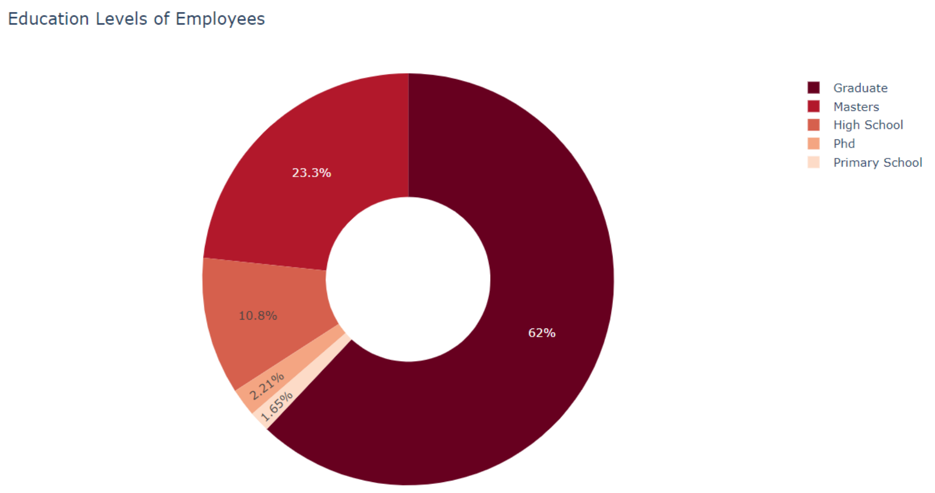
```
[247] px.box(data_frame=df_train, x='target', y='city_development_index', color_discrete_sequence=px.colors.sequential.Rainbow)
```



The plot shows that if the index of the cities development is above 0.85, then the candidates may not change their jobs.

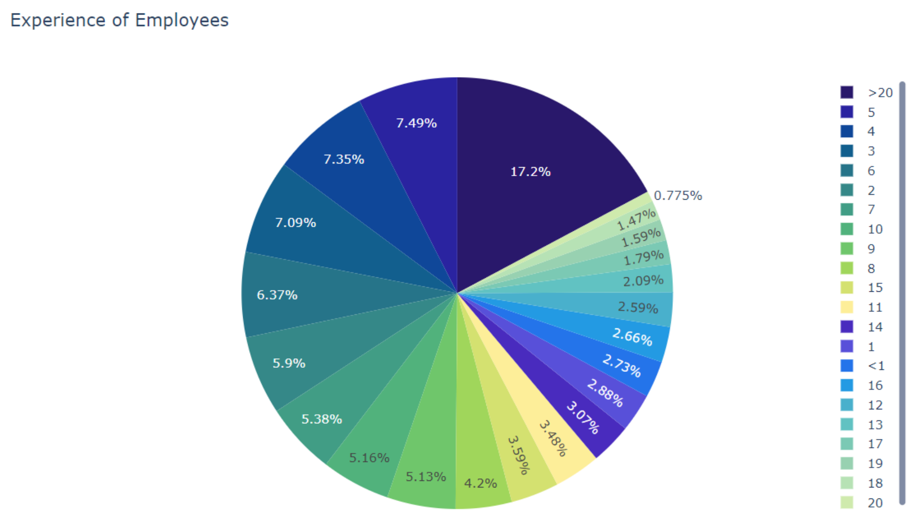
6. Piechart: education levels of employee

```
tmp = df_train['education_level'].value_counts().reset_index()
tmp.columns = ['education_level', 'percent']
tmp['percent']/=len(df_train)

fig = px.pie(tmp, names='education_level', values='percent', title='Education Levels of Employees',
             width=1000, height=600, color_discrete_sequence=px.colors.sequential.RdBu, hole=.4)

fig.show()
```



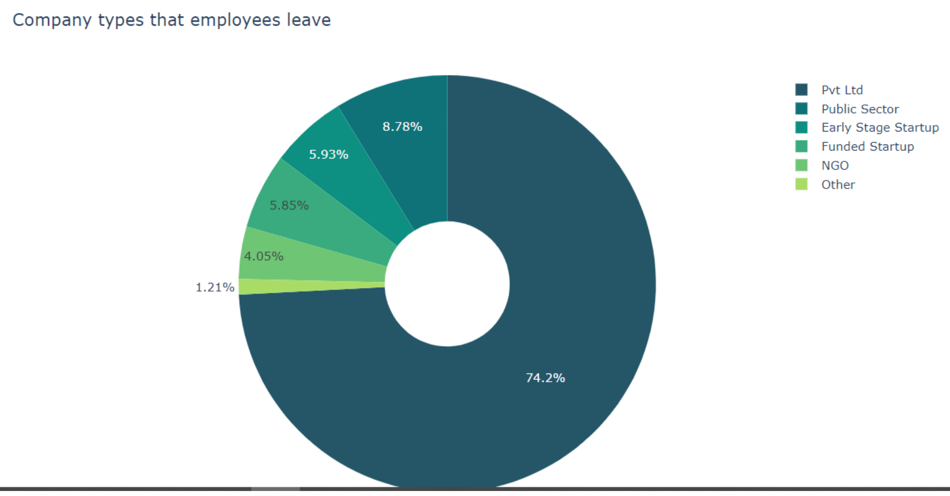
7. Piechart: Experience of employees

```
[249] tmp = df_train['experience'].value_counts().reset_index()
      tmp.columns = ['experience', 'percent']
      tmp['percent']/=len(df_train)

      fig = px.pie(tmp, names='experience', values='percent', title='Experience of Employees',
                   width=1000, height=600, color_discrete_sequence=px.colors.sequential.haline)

      fig.show()
```



8. Piechart: Company type

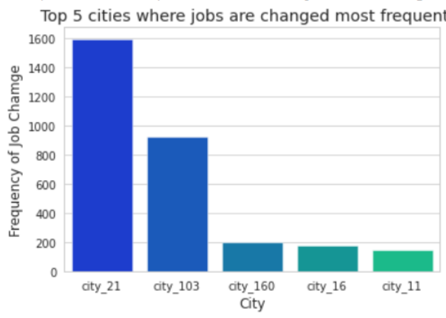
```
[250] company_type = df_train[df_train['target'] == 1]['company_type']
      tmp = company_type.value_counts().reset_index()
      tmp.columns = ['company_type', 'percent']
      tmp['percent']/=len(df_train)

      fig = px.pie(tmp, names='company_type', values='percent', title='Company types that employees leave',
                  width=1000, height=600, color_discrete_sequence=px.colors.sequential.Aggrnyl,
                  hole=0.3)

      fig.show()
```



9. Top cities related to job change

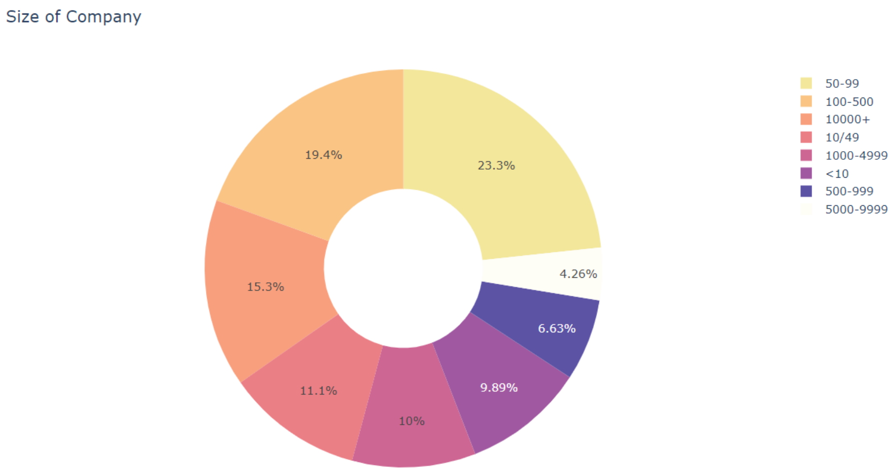
```
[251] city_frequency = list()
      unique_cities = df_train['city'].unique()

      for city_id, city in enumerate(unique_cities):
          temp = df_train[(df_train['city'] == city) & (df_train['target'] == 1.0)]
          frequency = temp.shape[0]
          city_frequency.append([city, frequency])

      city_data = pd.DataFrame(city_frequency, columns = ['city_name', 'frequency'])
      sorted_city_frequency = city_data.sort_values(by = 'frequency', ascending = False)

      top5 = sorted_city_frequency.iloc[:5, :]
      sns.barplot(x = 'city_name', y='frequency', data = top5, palette='winter')

      plt.xlabel('City')
      plt.ylabel('Frequency of Job Chamge')
      plt.title('Top 5 cities where jobs are changed most frequently')
```

```
Text(0.5, 1.0, 'Top 5 cities where jobs are changed most frequently')
```

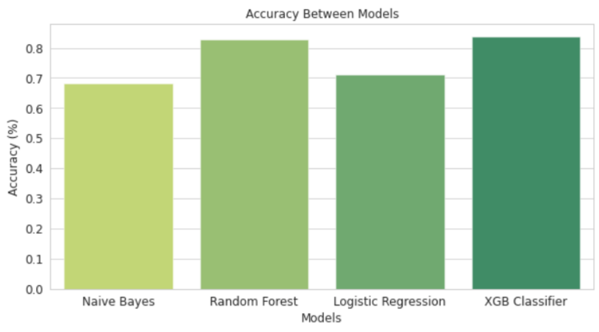


10. Piechart: Impact of company size

```
[252] tmp = df_train['company_size'].value_counts().reset_index()
      tmp.columns = ['company_size', 'percent']
      tmp['percent']/=len(df_train)

      fig = px.pie(tmp, names='company_size', values='percent', title='Size of Company',
                   width=1000, height=600, color_discrete_sequence=px.colors.sequential.Sunset,
                   hole=.4)

      fig.show()
```



```
[270] model_comparison['model'] = ['Naive Bayes', 'Random Forest', 'Logistic Regression', 'XGB Classifier']

      fig, ax = plt.subplots(figsize=(10,5))
      ax = sns.barplot('model', 'mean_test_accuracy', data=model_comparison, capsize=.05, palette='summer_r', ci=None)
      ax.set_xlabel("Models",fontsize=12)
      ax.set_ylabel("Accuracy (%)",fontsize=12)
      ax.tick_params(labelsize=12)
      ax.axes.set_title("Accuracy Between Models", fontsize=12)

      plt.show()
```


Accuracy Between Models
Accuracy (%)
Naive Bayes
Random Forest
Logistic Regression
XGB Classifier
Models