```python
In [1]: import pandas as pd
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
%matplotlib inline
```

Exploratory data analysis

Now we Read the data and fine tune any missing values in it and fill them using different techniques and remove certain rows if necessary.

```python
In [2]: data=pd.read_excel("job evaluation survey.xlsx")
```

Changing the column names for conventional purpose

```python
In [3]: data=data.rename(columns={'Which of the following describes the department that you work in?':'Department'})
data=data.rename(columns={'Since how long have you been engaged with the company':'Experience'})
data=data.rename(columns={"How satisfied are you with company, considering all the aspects?":'Satisfaction'})
data=data.rename(columns={"Which of the following describes your work position in the firm?":"Position"})
data=data.rename(columns={"On a scale of 1 to 5 how much satisfied are you with the income received? ( 5 being very much satisfied and 1 being least satisfied )":"income_rating"})
```

```python
In [4]: data=data.rename(columns={"Was there an increase in the work load while working from home ? ( during the pandemic period )":"Workload"})
data=data.rename(columns={"Were there extra benefits provided by the company during the covid time ?":"Benefits"})
data=data.rename(columns={"Does your company  give enough  opportunities for your career growth in the company?":"Oppurutinities"})
data=data.rename(columns={"How often do you feel stressed at work ?":"Stress"})
```

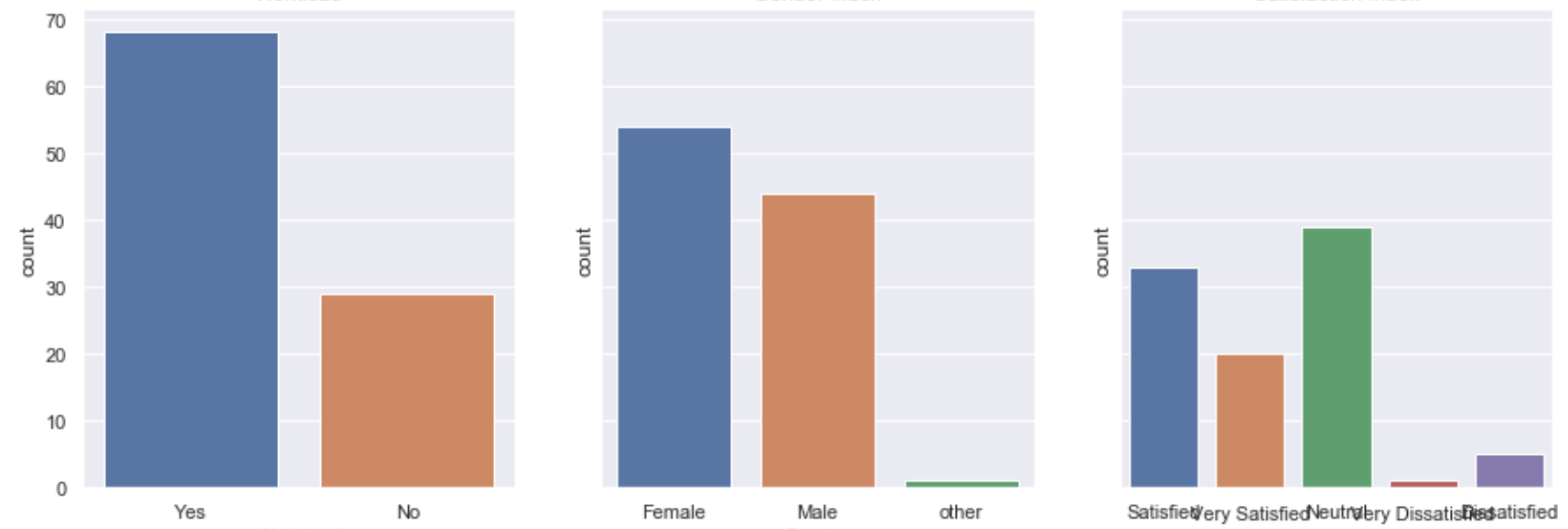
```python
In [5]: data=data.rename(columns={"How likely are you to look for another job outside the company ?":"Looking_newjob"})
data=data.rename(columns={"On a scale of 1 to 10, rate your experience with the company till date?":"rating"})
```

```python
In [6]: data=data.rename(columns={"Have you ever observed or faced any sort of the following discrimination or harassment at your work place ?":"Harassment"})
data=data.rename(columns={"On a scale of 1 to 5 how does the Management and Managers help in building your personality ? ( 5 being very much satisfied and 1 being least satisfied)":"Manager_rating"})
```

```python
In [7]: #dropping the timestamp as all are filled on 2021 and in time gap of few days
data.drop("Timestamp",axis=1,inplace=True)
```

```python
In [8]: #plotting the various relations between the categories in the data
fig,axes=plt.subplots(1,3,figsize=(15, 5), sharey=True)
sns.countplot(data["Satisfaction"],ax=axes[2])
axes[2].set_title("Satisfaction Index")
sns.countplot(data["Gender"],ax=axes[1])
axes[1].set_title("Gender Index")
sns.countplot(data["Workload"],ax=axes[0])
axes[0].set_title("Workload")
```

```
Out[8]: Text(0.5, 1.0, 'Workload')
```




```python
In [9]: #considering the not likely as unlikely itself and changing the values in a better way
data["Looking_newjob"]=data["Looking_newjob"].map({'Not sure':'Unlikely','Most Likely':'Likely','Most likely':'Likely'})
```

```python
In [10]: #filling the null values as unlikely in looking for new job column
data["Looking_newjob"].fillna(value='Unlikely',inplace=True)
```

```python
In [11]: data["Workload"]=data["Workload"].map({'Yes':1,'No':0})
```

```python
In [12]: data["Looking_newjob"]=data["Looking_newjob"].map({'Likely':'1','Unlikely':'0'})
```

```python
In [13]: data["Looking_newjob"]=data["Looking_newjob"].astype(int)
```

```python
In [14]: #You can figure out the number of people looking for new job in different departments

fig_dims = (14, 4)
fig, ax = plt.subplots(figsize=fig_dims)

sns.barplot(x=data["Department"],y=data["Looking_newjob"],ax=ax,data=data,hue=data["Gender"])
```

```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x1ff9d3d9e48>
```


```python
In [15]: data["Oppurutinities"]=data["Oppurutinities"].map({'Yes':1,'No':0,'Maybe':1})
```

```python
In [16]: data["Benefits"]=data["Benefits"].map({'Yes':1,'No':0,'Maybe':1})
```

```python
In [17]: data["Stress"].unique()
```

```
Out[17]: array(['Sometimes', 'Often', 'Never', 'Rarely', nan, 'Always'],
      dtype=object)
```

```python
In [18]: #creating a function to change the values in stress column

def stress(x):
    if x=='Never':
        return 0
    else:
        return 1
```

```python
In [19]: data["Stress"]=data["Stress"].map(stress)
```

```python
In [20]: data["Age"].unique()
```

```
Out[20]: array(['20-30', nan, '30-40', '40-50', '50+'], dtype=object)
```

```python
In [21]: data["Age"]=data["Age"].fillna('25+')
```

```python
In [22]: def age(x):
    if '-' in x:
        a=x.split('-')
        a1=list(map(int,a))
        avg=(a1[0]+a1[1])//2
        return avg
    else:
        a=x.split('+')
        a1=int(a[0])
        return a1
```

```python
In [23]: data["Age"]=data["Age"].apply(age)
```

```python
In [24]: data["Experience"]=data["Experience"].map({'1-2 Years':1.5,'0-6 Months':0.6,'2-3 Years':2.5,'3-4 Years':3.5,'More than 5 Years':5})
```

```python
In [25]: data["Gender"]=data["Gender"].map({'Male':0,'Female':1,'other':1})
```

```python
In [26]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             100 non-null    int64
 1   Gender          99 non-null     float64
 2   Department      98 non-null     object
 3   Experience      95 non-null     float64
 4   Satisfaction    98 non-null     object
 5   Position        85 non-null     object
 6   income_rating   98 non-null     float64
 7   Workload        97 non-null     float64
 8   Benefits        66 non-null     float64
 9   Harassment      74 non-null     object
 10  Oppurutinities  98 non-null     float64
 11  Manager_rating  97 non-null     float64
 12  Stress          100 non-null    int64
 13  Looking_newjob  100 non-null    int32
 14  rating          98 non-null     float64
dtypes: float64(8), int32(1), int64(2), object(4)
memory usage: 11.5+ KB
```

```python
In [27]: data["Benefits"].unique()
```

```
Out[27]: array([nan,  0.,  1.])
```

```python
In [28]: data["Benefits"]=data["Benefits"].fillna(0)
```

```python
In [29]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             100 non-null    int64
 1   Gender          99 non-null     float64
 2   Department      98 non-null     object
 3   Experience      95 non-null     float64
 4   Satisfaction    98 non-null     object
 5   Position        85 non-null     object
 6   income_rating   98 non-null     float64
 7   Workload        97 non-null     float64
 8   Benefits        100 non-null    float64
 9   Harassment      74 non-null     object
 10  Oppurutinities  98 non-null     float64
 11  Manager_rating  97 non-null     float64
 12  Stress          100 non-null    int64
 13  Looking_newjob  100 non-null    int32
 14  rating          98 non-null     float64
dtypes: float64(8), int32(1), int64(2), object(4)
memory usage: 11.5+ KB
```

```python
In [30]: data["Gender"]=data["Gender"].fillna(1)
data["Workload"]=data["Workload"].fillna(0)
data["Harassment"]=data["Harassment"].fillna(0)
```

```python
In [34]: data["income_rating"]=data["income_rating"].fillna(data["income_rating"].mode()[0])
```

```python
In [35]: data["Manager_rating"]=data["Manager_rating"].fillna(data["Manager_rating"].mode()[0])
```

```python
In [36]: data["rating"]=data["rating"].fillna(data["rating"].mode()[0])
```

```python
In [37]: data["Oppurutinities"]=data["Oppurutinities"].fillna(0)
```

```python
In [38]: data.head()
```

Out[38]:

	Age	Gender	Department	Experience	Satisfaction	Position	income_rating	Workload	Benefits	Harassment	Oppurutinities	Mana
0	25	1.0	Customer support	1.5	Satisfied	IT consultant	3.0	1.0	0.0	Suppression by the Management	0.0	
1	25	0.0	Business development	0.6	Satisfied	Business Analyst	5.0	1.0	0.0	Racial discrimination, Age discrimination	1.0	
2	25	1.0	Test team	0.6	Very Satisfied	Business Analyst	5.0	1.0	1.0	Age discrimination	1.0	
3	25	1.0	other	0.6	Satisfied	Project Manager	3.0	1.0	1.0	Gender Bias	1.0	
4	25	1.0	Customer support	1.5	Satisfied	IT consultant	1.0	1.0	0.0	Racial discrimination	1.0	

```python
In [39]: data["Satisfaction"]=data["Satisfaction"].map({'Satisfied':1,"Very Satisfied":1,'Dissatisfied':0,'Neutral':1})
```

```python
In [40]: data["Satisfaction"]=data["Satisfaction"].fillna(0)
```

Machine Learning Model

There are two types of variables : 1)Independent variable 2)Dependent variable We will find the dependent variable with the help of independent variables if there exists a linear relationship between both dependent and independent variable

our machine learning model predicts whether a person will look for new job or not based on various inputs

```python
In [54]: features=['Age', 'Gender', 'Experience', 'Satisfaction','income_rating', 'Workload', 'Benefits', 'Harassment', 'Oppurutinities',
       'Manager_rating', 'Stress', 'rating']
```

```python
In [55]: #X is a independent variable
#Y is dependent variable


X=data[features]
y=data["Looking_newjob"]
```

```python
In [51]: def discrimination(x):
    if x=='No'or x=='Never' or x=='Non' or x=='Lower Package' or x=='No ' or x=="Nothing":
        return 0
    else:
        return 1
```

```python
In [52]: data["Harassment"]=data["Harassment"].apply(discrimination)
```

```python
In [53]: data["Experience"]=data["Experience"].fillna(data["Experience"].mode()[0])
```

SPLITTING THE DATA:

we will split the given data into training set and testing set

Training set is used to train the model

Testing set is used to the test the model

```python
In [56]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
```

SUPERVISED MACHINE LEARNING ALGORITHM

```python
In [57]: from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

```python
In [58]: lr = LogisticRegression(solver='liblinear',multi_class='ovr')
lr.fit(X_train, y_train)
lr.score(X_test, y_test)
```

```
Out[58]: 0.7
```

```python
In [59]: sv=SVC(gamma="auto")
sv.fit(X_train, y_train)
sv.score(X_test, y_test)
```

```
Out[59]: 0.7
```

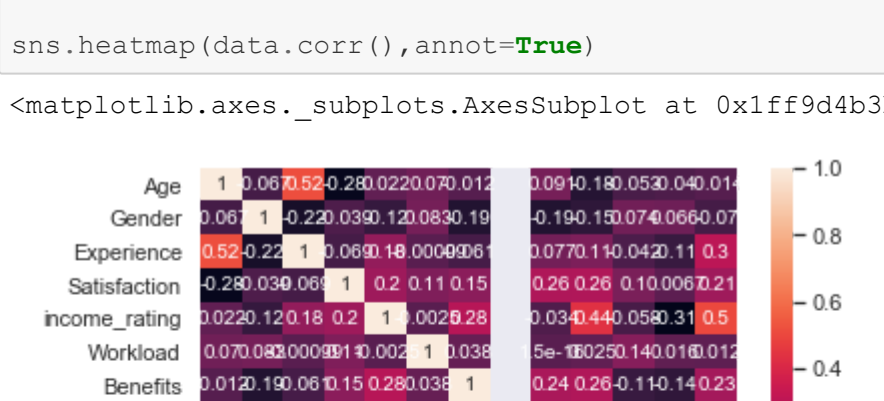
```python
In [60]: rf = RandomForestClassifier(n_estimators=40)
rf.fit(X_train, y_train)
rf.score(X_test, y_test)
```

```
Out[60]: 0.7
```

```python
In [61]: #Finding the correlation between the variables in the data

sns.heatmap(data.corr(),annot=True)
```

```
Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x1ff9d4b3b48>
```




```python
In [62]: from sklearn.model_selection import cross_val_score
```

```python
In [63]: cross_val_score(LogisticRegression(solver='liblinear',multi_class='ovr'),X,y,cv=3)
```

```
Out[63]: array([0.73529412, 0.66666667, 0.78787879])
```

```python
In [64]: cross_val_score(SVC(gamma='auto'), X,y,cv=3)
```

```
Out[64]: array([0.73529412, 0.75757576, 0.72727273])
```

```python
In [65]: cross_val_score(RandomForestClassifier(n_estimators=40),X,y,cv=3)
```

```
Out[65]: array([0.79411765, 0.66666667, 0.78787879])
```

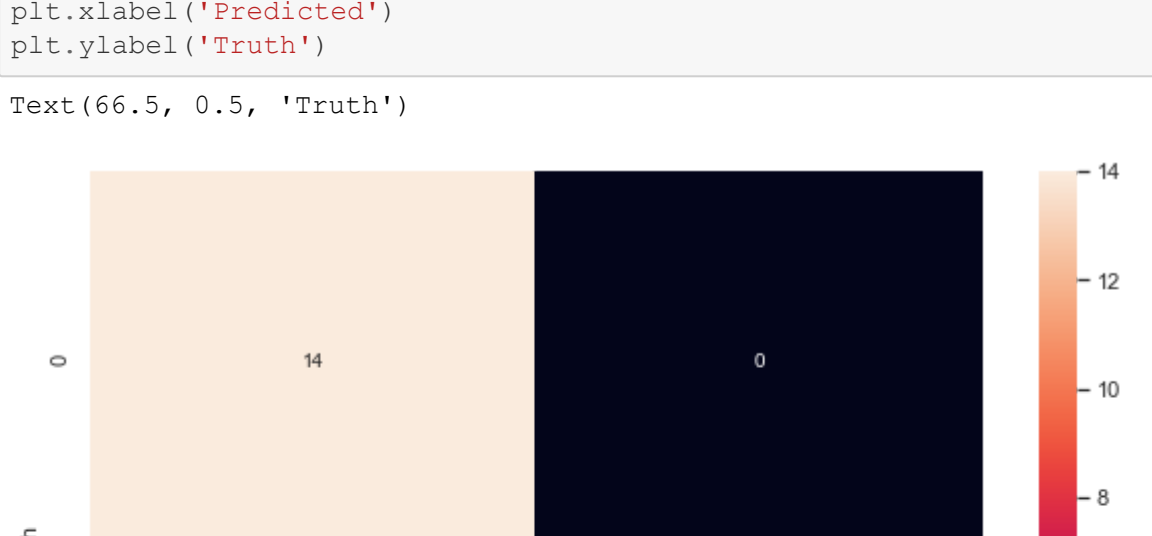
```python
In [66]: from sklearn.metrics import confusion_matrix
```

```python
In [67]: y_pred=lr.predict(X_test)
```

```python
In [68]: c=confusion_matrix(y_test,y_pred)
```

```python
In [69]: import seaborn as sn
plt.figure(figsize=(10,7))
sn.heatmap(c, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Out[69]: Text(66.5, 0.5, 'Truth')
```




```python
In [70]: fig_dims = (14, 4)
fig, ax = plt.subplots(figsize=fig_dims)

sns.barplot(x=data["Department"],y=data["Looking_newjob"],ax=ax,data=data,hue=data["Gender"])
```

```
Out[70]: <matplotlib.axes._subplots.AxesSubplot at 0x1ff9e50d208>
```


```python
In [71]: data.head()
```

Out[71]:

	Age	Gender	Department	Experience	Satisfaction	Position	income_rating	Workload	Benefits	Harassment	Oppurutinities	Manag
0	25	1.0	Customer support	1.5	1.0	IT consultant	3.0	1.0	0.0	1	0.0	
1	25	0.0	Business development	0.6	1.0	Business Analyst	5.0	1.0	0.0	1	1.0	
2	25	1.0	Test team	0.6	1.0	Business Analyst	5.0	1.0	1.0	1	1.0	
3	25	1.0	other	0.6	1.0	Project Manager	3.0	1.0	1.0	1	1.0	
4	25	1.0	Customer support	1.5	1.0	IT consultant	1.0	1.0	0.0	1	1.0	

```python
In [72]: #Fine tunning the model with the various parameters

grid_param = {
    "n_estimators" : [10],
    'criterion': ['gini', 'entropy'],
    'max_depth' : range(2,20,1),
    'min_samples_leaf' : range(1,6,1),
    'min_samples_split': range(2,6,1),
    'max_features' : ['auto','log2']
}
```

```python
In [73]: from sklearn.model_selection import GridSearchCV
```

```python
In [74]: grid_search = GridSearchCV(estimator=rf,param_grid=grid_param,cv=2,n_jobs =-1,verbose = 3)
```

```python
In [75]: grid_search.fit(X_train,y_train)
```

```
Fitting 2 folds for each of 1440 candidates, totalling 2880 fits
```

```
Out[75]: GridSearchCV(cv=2, estimator=RandomForestClassifier(n_estimators=40), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 20),
                         'max_features': ['auto', 'log2'],
                         'min_samples_leaf': range(1, 6),
                         'min_samples_split': range(2, 6),
                         'n_estimators': [10]},
             verbose=3)
```

```python
In [76]: grid_search.best_params_
```

```
Out[76]: {'criterion': 'entropy',
 'max_depth': 14,
 'max_features': 'auto',
 'min_samples_leaf': 1,
 'min_samples_split': 4,
 'n_estimators': 10}
```

```python
In [78]: rf = RandomForestClassifier(criterion= 'entropy',
 max_depth= 14,
 max_features= 'auto',
 min_samples_leaf= 1,
 min_samples_split= 4,
 n_estimators= 10)
rf.fit(X_train, y_train)
rf.score(X_test, y_test)
```

```
Out[78]: 0.8
```

We have achieved a model with 80%accuracy that can predict whether a person will look for job change or not by providing various inputs.