Q.Create a classification model to find the value of type column: Ans:

Classification: Performing necessary EDA and pre-processing on given data and finding out which classification method is best for given dataset. steps are as follow:

In [1]:
```
#import necessary libraries:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

#read dataset:
df = pd.read_csv('Freelance Platform Projects.csv')

#display first five records of dataset:
df.head()
```

Out[1]:

	Title	Category Name	Experience	Sub Category Name	Currency	Budget	Location	Freelancer Preferred From	Type	Date Posted	Description	Duration	Client Registration Date	Client City	Client Country	Client Currency	Client Job Title
0	Banner images for web design websites	Design	Entry ($)	Graphic Design	EUR	60.0	remote	ALL	fixed_price	2023-04-29 18:06:39	We are looking to improve the banner images on...	NaN	2010-11-03	Dublin	Ireland	EUR	PPC Management
1	Make my picture a solid silhouette	Video, Photo & Image	Entry ($)	Image Editing	GBP	20.0	remote	ALL	fixed_price	2023-04-29 17:40:28	Hello Vinit need a quick designer to make 4 pi...	NaN	2017-02-21	London	United Kingdom	GBP	Office manager
2	Bookkeeper needed	Business	Entry ($)	Finance & Accounting	GBP	12.0	remote	ALL	fixed_price	2023-04-29 17:40:06	Hi - I need a bookkeeper to assist with bookke...	NaN	2023-04-09	London	United Kingdom	GBP	Paralegal
3	Accountant needed	Business	Entry ($)	Tax Consulting & Advising	GBP	14.0	remote	ALL	fixed_price	2023-04-29 17:32:01	Hi - I need an accountant to assist me with un...	NaN	2023-04-09	London	United Kingdom	GBP	Paralegal
4	Guest Post on High DA Website	Digital Marketing	Expert ($$$)	SEO	USD	10000.0	remote	ALL	fixed_price	2023-04-29 17:09:36	Hi, I am currently running a project where I w...	NaN	2016-07-01	Mumbai	India	USD	Guest posts buyer

In [2]:
```
#display rows and columns of adtaset respectively(rows,columns):
df.shape
```

Out[2]:
```
(12222, 17)
```

In [3]:
```
#display information of data set:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12222 entries, 0 to 12221
Data columns (total 17 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Title                     12222 non-null  object
 1   Category Name             12222 non-null  object
 2   Experience                12222 non-null  object
 3   Sub Category Name         12222 non-null  object
 4   Currency                  12222 non-null  object
 5   Budget                    12222 non-null  float64
 6   Location                  12222 non-null  object
 7   Freelancer Preferred From  12222 non-null  object
 8   Type                      12222 non-null  object
 9   Date Posted               12222 non-null  object
 10  Description               12222 non-null  object
 11  Duration                  1602 non-null   object
 12  Client Registration Date  12222 non-null  object
 13  Client City               12222 non-null  object
 14  Client Country            12222 non-null  object
 15  Client Currency           12222 non-null  object
 16  Client Job Title          4588 non-null   object
dtypes: float64(1), object(16)
memory usage: 1.6+ MB
```

EDA and Preprocessing:

In [4]:
```
#identifying null values:
df.isna().sum()
```

Out[4]:
```
Title                          0
Category Name                  0
Experience                     0
Sub Category Name              0
Currency                       0
Budget                         0
Location                       0
Freelancer Preferred From      0
Type                           0
Date Posted                    0
Description                    0
Duration                   10620
Client Registration Date       0
Client City                    0
Client Country                 0
Client Currency                0
Client Job Title            7634
dtype: int64
```

In [5]:
```
#display datatypes of columns:
df.dtypes
```

Out[5]:
```
Title                         object
Category Name                 object
Experience                    object
Sub Category Name             object
Currency                      object
Budget                       float64
Location                      object
Freelancer Preferred From     object
Type                          object
Date Posted                   object
Description                   object
Duration                      object
Client Registration Date      object
Client City                   object
Client Country                object
Client Currency               object
Client Job Title              object
```

In [6]:
```
#changing datatype of columns using labelencoder:
encode = ['Title','Category Name','Sub Category Name','Freelancer Preferred From','Description',
                'Client City','Client Country','Type','Date Posted','Duration','Client Job Title','Currency','Location','Experience','Client Registration Date','Client Currency']

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for column in encode:
    df[column] = le.fit_transform(df[column])
```

In [7]:
```
#display datatypes of column
df.dtypes
```

Out[7]:
```
Title                         int32
Category Name                 int32
Experience                    int32
Sub Category Name             int32
Currency                      int32
Budget                      float64
Location                      int32
Freelancer Preferred From     int32
Type                          int32
Date Posted                   int32
Description                   int32
Duration                      int32
Client Registration Date      int32
Client City                   int32
Client Country                int32
Client Currency               int32
Client Job Title              int32
dtype: object
```

In [8]:
```
#dropping unnecessary columns:
df.drop(columns=['Duration','Client Job Title'],inplace=True)
```

In [9]:
```
#display columns of dataset:
df.columns
```

Out[9]:
```
Index(['Title', 'Category Name', 'Experience', 'Sub Category Name', 'Currency',
       'Budget', 'Location', 'Freelancer Preferred From', 'Type',
       'Date Posted', 'Description', 'Client Registration Date', 'Client City',
       'Client Country', 'Client Currency'],
      dtype='object')
```

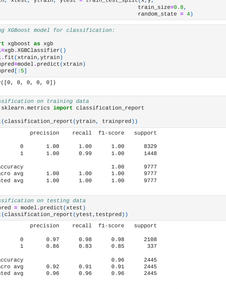
In [10]:
```
#defining variable x and variable y respectively:

# variable x contaion all columns except column Type:
x = df.drop(columns=['Type'])

# variable Y contaion only one column X:
y = df['Type']
```

In [11]:
```
#checking value counts of target variable Y:
y.value_counts()
```

Out[11]:
```
0    10437
1     1785
Name: Type, dtype: int64
```

In [22]:
```
#identifying outliers and Handling:
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(df['Budget'])
```

Out[22]:
```
<Axes: >
```



In [24]:
```
#handling skewness:
df['Budget'].skew()
```

Out[24]:
```
42.455398395555996
```

In [27]:
```
#handling skewness:
from scipy.stats import boxcox
df['Budget'] = df['Budget']+1
df['Budget'] = boxcox(df['Budget'],+1)[0]
df['Budget'].skew()
```

Out[27]:
```
0.006936377365312578
```

In [28]:
```
#plotting histogram:
plt.hist(df['Budget'])
plt.show()
```


In [29]:
```
#boxplot
sns.boxplot(df['Budget'])
```

Out[29]:
```
<Axes: >
```


In [42]:
```
#dropping the outliers from Budget_usd
df=df.drop(df[df['Budget'] >5.2].index)
df=df.drop(df[df['Budget']<2].index)
```

In [43]:
```
#boxplot after removing outliers
sns.boxplot(df['Budget'])
plt.show()
```


In [48]:
```
#encoding column type that is target variable y:

from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
y = encoder.fit_transform(y)
```

Classification is as follows:

In [49]:
```
#spliting the data into training and testing:
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x,y,
                                                train_size=0.8,
                                                random_state = 4)
```

In [50]:
```
#Using XGBoost model for classification:

import xgboost as xgb
model=xgb.XGBClassifier()
model.fit(xtrain,ytrain)
trainpred=model.predict(xtrain)
trainpred[:5]
```

Out[50]:
```
array([0, 0, 0, 0, 0])
```

In [51]:
```
#classification on training data
from sklearn.metrics import classification_report

print(classification_report(ytrain, trainpred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      8329
           1       1.00      0.99      1.00      1448

    accuracy                           1.00      9777
   macro avg       1.00      1.00      1.00      9777
weighted avg       1.00      1.00      1.00      9777
```

In [52]:
```
#classification on testing data
testpred = model.predict(xtest)
print(classification_report(ytest,testpred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.98      0.98      2108
           1       0.86      0.83      0.85       337

    accuracy                           0.96      2445
   macro avg       0.92      0.91      0.91      2445
weighted avg       0.96      0.96      0.96      2445
```

Observation: From above calculation we can conclude that XGboost model is suitable for classification as it gives better accuracy compared to other models.