```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

from sklearn import linear_model
from scipy.stats import describe
from sklearn import preprocessing
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_score
from sklearn.metrics import f1_score, confusion_matrix, precision_recall_curve, roc_curve
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

df = pd.read_csv('data.csv')
df.head()
```

	Gender	Home Location	Level of Education	Age(Years)	Number of Subjects	Device type used to attend classes	Economic status	Family size	Interr facil in yc local
0	Male	Urban	Under Graduate	18	11	Laptop	Middle Class	4	
1	Male	Urban	Under Graduate	19	7	Laptop	Middle Class	4	
2	Male	Rural	Under Graduate	18	5	Laptop	Middle Class	5	
3	Male	Urban	Under Graduate	18	5	Laptop	Middle Class	4	
4	Male	Rural	Under Graduate	18	5	Laptop	Middle Class	4	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1033 entries, 0 to 1032
Data columns (total 23 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   Gender                                    1033 non-null   object
 1   Home Location                             1033 non-null   object
```

```
 1   Home Location                                                  1033 non-null   object
 2   Level of Education                                             1033 non-null   object
 3   Age(Years)                                                     1033 non-null   int64
 4   Number of Subjects                                             1033 non-null   int64
 5   Device type used to attend classes                             1033 non-null   object
 6   Economic status                                                1033 non-null   object
 7   Family size                                                    1033 non-null   int64
 8   Internet facility in your locality                             1033 non-null   int64
 9   Are you involved in any sports?                                1033 non-null   object
 10  Do elderly people monitor you?                                 1033 non-null   object
 11  Study time (Hours)                                             1033 non-null   int64
 12  Sleep time (Hours)                                             1033 non-null   int64
 13  Time spent on social media (Hours)                             1033 non-null   int64
 14  Interested in Gaming?                                          1033 non-null   object
 15  Have separate room for studying?                               1033 non-null   object
 16  Engaged in group studies?                                      1033 non-null   object
 17  Average marks scored before pandemic in traditional classroom  1033 non-null   object
 18  Your interaction in online mode                                1033 non-null   int64
 19  Clearing doubts with faculties in online mode                  1033 non-null   int64
 20  Interested in?                                                 1033 non-null   object
 21  Performance in online                                          1033 non-null   int64
 22  Your level of satisfaction in Online Education                 1033 non-null   object
dtypes: int64(10), object(13)
memory usage: 185.7+ KB
```

```
df.shape
```

```
(1033, 23)
```

```
df = df[['Home Location', 'Level of Education', 'Number of Subjects', 'Device type used to attend classes',
       'Economic status', 'Internet facility in your locality', 'Are you involved in any sports?', 'Do elderly peop
       'Study time (Hours)', 'Sleep time (Hours)', 'Time spent on social media (Hours)', 'Interested in Gaming?
       'Engaged in group studies?', 'Average marks scored before pandemic in traditional classroom',
       'Your interaction in online mode', 'Clearing doubts with faculties in online mode',
       'Interested in?', 'Performance in online', 'Your level of satisfaction in Online Education']]
df.head()
```

		Number	Device type		Internet	Are you	Do elderly	Study

Home Location	Level of Education	Number of Subjects	used to	Economic status	facility in your	involved in any	elderly people monitor	Study time (Hours)

```
encoder = LabelEncoder()
encoder.fit(df['Home Location'])
df['Home Location'] = encoder.transform(df['Home Location'])

encoder = LabelEncoder()
encoder.fit(df['Level of Education'])
df['Level of Education'] = encoder.transform(df['Level of Education'])

encoder = LabelEncoder()
encoder.fit(df['Device type used to attend classes'])
df['Device type used to attend classes'] = encoder.transform(df['Device type used to attend classes'])

encoder = LabelEncoder()
encoder.fit(df['Economic status'])
df['Economic status'] = encoder.transform(df['Economic status'])

encoder = LabelEncoder()
encoder.fit(df['Are you involved in any sports?'])
df['Are you involved in any sports?'] = encoder.transform(df['Are you involved in any sports?'])

encoder = LabelEncoder()
encoder.fit(df['Do elderly people monitor you?'])
df['Do elderly people monitor you?'] = encoder.transform(df['Do elderly people monitor you?'])

encoder = LabelEncoder()
encoder.fit(df['Interested in Gaming?'])
df['Interested in Gaming?'] = encoder.transform(df['Interested in Gaming?'])

encoder = LabelEncoder()
encoder.fit(df['Engaged in group studies?'])
df['Engaged in group studies?'] = encoder.transform(df['Engaged in group studies?'])

encoder = LabelEncoder()
encoder.fit(df['Average marks scored before pandemic in traditional classroom'])
df['Average marks scored before pandemic in traditional classroom'] = encoder.transform(df['Average marks

encoder = LabelEncoder()
encoder.fit(df['Interested in?'])
df['Interested in?'] = encoder.transform(df['Interested in?'])

encoder = LabelEncoder()
encoder.fit(df['Your level of satisfaction in Online Education'])
df['Your level of satisfaction in Online Education'] = encoder.transform(df['Your level of satisfaction in Online


df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1033 entries, 0 to 1032

Data columns (total 19 columns):
 #   Column                                              Non-Null Count  Dtype
---  ------                                              --------------  -----
 0   Home Location                                       1033 non-null   int64
```

```
 1   Level of Education                                                1033 non-null   int64
 2   Number of Subjects                                                1033 non-null   int64
 3   Device type used to attend classes                                1033 non-null   int64
 4   Economic status                                                   1033 non-null   int64
 5   Internet facility in your locality                                1033 non-null   int64
 6   Are you involved in any sports?                                   1033 non-null   int64
 7   Do elderly people monitor you?                                    1033 non-null   int64
 8   Study time (Hours)                                                1033 non-null   int64
 9   Sleep time (Hours)                                                1033 non-null   int64
 10  Time spent on social media (Hours)                                1033 non-null   int64
 11  Interested in Gaming?                                             1033 non-null   int64
 12  Engaged in group studies?                                         1033 non-null   int64
 13  Average marks scored before pandemic in traditional classroom     1033 non-null   int64
 14  Your interaction in online mode                                   1033 non-null   int64
 15  Clearing doubts with faculties in online mode                     1033 non-null   int64
 16  Interested in?                                                    1033 non-null   int64
 17  Performance in online                                             1033 non-null   int64
 18  Your level of satisfaction in Online Education                    1033 non-null   int64
dtypes: int64(19)
memory usage: 153.5 KB
```

```
df = df.rename(columns={'Home Location' : 'home', 'Level of Education' : 'education',
                        'Device type used to attend classes' : 'device', 'Economic status' : 'economic',
                        'Do elderly people monitor you?' : 'monitoring', 'Study time (Hours)' : 'studytime',
                        'Sleep time (Hours)' : 'sleeptime', 'Time spent on social media (Hours)' : 'socialmedia',
                        'Internet facility in your locality' : 'internet', 'Are you involved in any sports?' : 'sports',
                        'Interested in Gaming?': 'game', 'Engaged in group studies?' : 'groupstudies',
                        'Average marks scored before pandemic in traditional classroom' : 'averagescored',
                        'Your interaction in online mode' : 'interaction', 'Clearing doubts with faculties in online mo
                        'Interested in?' : 'interesting', 'Performance in online' :'performance',
                        'Your level of satisfaction in Online Education' :'satisfaction' })
df
```

	home	education	Number of Subjects	device	economic	internet	sports	monitoring	studyti

```
# 확인필요
x = df.drop(labels = ["satisfaction"], axis=1)
y = df['satisfaction']


# split data into training and test
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)

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(826, 18)
(207, 18)
(826,)
(207,)
```

1032 1 2 6 1 0 3 0 0

```
from sklearn.linear_model import LinearRegression
lin_model = LinearRegression()

# fit the model to the training data
lin_model_fit = lin_model.fit(x_train, y_train)

# predict the data
predict = lin_model_fit.predict(x_test)

# calculate RMSE (root mean square error) and R^2 (predictive power)
from sklearn.metrics import mean_squared_error, r2_score
rmse = (np.sqrt(mean_squared_error(y_test, predict)))
r2 = r2_score(y_test, predict)

# print the performance metrics
print("Model performance")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("₩n")
```

```
Model performance
--------------------------------------
RMSE is 0.8118258273902274
R2 score is 0.02492534198294738
```

L1 Regularisation (LASSO)

```
from sklearn.linear_model import Lasso

model = Lasso(alpha=0.5)

# fit the model to the training data
model_fit = model.fit(x_train, y_train)
```

```
model_fit = model.fit(x_train, y_train)

# predict the data
predict = model_fit.predict(x_test)

# calculate RMSE (root mean square error) and R^2 (predictive power)
from sklearn.metrics import mean_squared_error, r2_score
rmse = (np.sqrt(mean_squared_error(y_test, predict)))
r2 = r2_score(y_test, predict)

# print the performance metrics
print("Model performance")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("₩n")
```

```
Model performance
--------------------------------------
RMSE is 0.8225118131290654
R2 score is -0.0009132304257932766
```

```
# print the beta values of the model (co-efficients)
betas = lin_model_fit.coef_
counter = 0
for col in df.columns:
    if counter == 0:
        print("Beta weights/co-efficients - UNREGULARISED")
        print("----------------------------------------")
    print(col + ": " + str(round(betas[counter], 4)))
    counter +=1

print("₩n")

# print the beta values of the model (co-efficients)
betas_1 = model_fit.coef_
counter = 0
for col in df.columns:
    if counter == 0:
        print("Beta weights/co-efficients - LASSO")
        print("----------------------------------------")
    print(col + ": " + str(round(betas_1[counter], 4)))
    counter +=1
```

```
Beta weights/co-efficients - UNREGULARISED
----------------------------------------
home: -0.0262
```

```
education: 0.0016
Number of Subjects: 0.0079
device: 0.063
economic: 0.1121
internet: 0.0113
sports: 0.0029
monitoring: -0.0604
studytime: 0.0082
sleeptime: -0.018
socialmedia: 0.0036
game: 0.025
groupstudies: 0.0235
averagescored: -0.0581
interaction: 0.0501
dobuts: 0.0377
interesting: -0.0133
performance: 0.066
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-19-231c8376d98a> in <module>()
      6         print("Beta weights/co-efficients - UNREGULARISED")
      7         print("----------------------------------------")
----> 8     print(col + ": " + str(round(betas[counter], 4)))
      9     counter +=1
```

```
from sklearn.linear_model import Ridge
l2_model = Ridge(alpha=0.5)

# fit the model to the training data
l2_model_fit = l2_model.fit(x_train, y_train)

# predict the data
predict = l2_model_fit.predict(x_test)

# calculate RMSE (root mean square error) and R^2 (predictive power)
from sklearn.metrics import mean_squared_error, r2_score
rmse = (np.sqrt(mean_squared_error(y_test, predict)))
r2 = r2_score(y_test, predict)

# print the performance metrics
print("Model performance")
print("-------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("₩n")
```

```
Model performance
-------------------------------------
RMSE is 0.8118047195358294
R2 score is 0.024976046125748996
```

```
# print the beta values of the model (co-efficients)

betas = lin_model_fit.coef_
counter = 0
for col in df.columns:
    if counter == 0:
```

```
    if counter == 0:
        print("Beta weights/co-efficients - UNREGULARISED")
        print("----------------------------------------")
    print(col + ": " + str(round(betas[counter], 4)))
    counter +=1

print("₩n")

# print the beta values of the model (co-efficients)
betas_l1 = l1_model_fit.coef_
counter = 0
for col in df.columns:
    if counter == 0:
        print("Beta weights/co-efficients - LASSO")
        print("----------------------------------------")
    print(col + ": " + str(round(betas_l1[counter], 4)))
    counter +=1

print("₩n")

# print the beta values of the model (co-efficients)
betas_l2 = l2_model_fit.coef_
counter = 0
for col in df.columns:
    if counter == 0:
        print("Beta weights/co-efficients - RIDGE")
        print("----------------------------------------")
    print(col + ": " + str(round(betas_l2[counter], 4)))
    counter +=1
```

```
Beta weights/co-efficients - UNREGULARISED
----------------------------------------
home: -0.0262
education: 0.0016
```

```
education: 0.0016
Number of Subjects: 0.0079
device: 0.063
economic: 0.1121
internet: 0.0113
sports: 0.0029
```

```
from sklearn.linear_model import ElasticNet
enet_model = ElasticNet(alpha=0.5, l1_ratio=0.5)

# fit the model to the training data
enet_model_fit = enet_model.fit(x_train, y_train)

# predict the data
predict = enet_model_fit.predict(x_test)

# calculate RMSE (root mean square error) and R^2 (predictive power)
from sklearn.metrics import mean_squared_error, r2_score
rmse = (np.sqrt(mean_squared_error(y_test, predict)))
r2 = r2_score(y_test, predict)

# print the performance metrics
print("Model performance")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("₩n")
```

```
Model performance
--------------------------------------
RMSE is 0.816315247184885
R2 score is 0.014111142123486076
```

```
# print the beta values of the model (co-efficients)
betas = lin_model_fit.coef_
counter = 0
for col in boston.columns:
    if counter == 0:
        print("Beta weights/co-efficients - UNREGULARISED")
        print("----------------------------------------")
    print(col + ": " + str(round(betas[counter], 4)))
    counter +=1

print("₩n")

# print the beta values of the model (co-efficients)
betas_l1 = l1_model_fit.coef_
counter = 0
for col in boston.columns:
    if counter == 0:

        print("Beta weights/co-efficients - LASSO")
        print("----------------------------------------")
    print(col + ": " + str(round(betas_l1[counter], 4)))
    counter += 1
```

```
    counter +=1

print("₩n")

# print the beta values of the model (co-efficients)
betas_l2 = l2_model_fit.coef_
counter = 0
for col in boston.columns:
    if counter == 0:
        print("Beta weights/co-efficients - RIDGE")
        print("----------------------------------------")
    print(col + ": " + str(round(betas_l2[counter], 4)))
    counter +=1

print("₩n")

# print the beta values of the model (co-efficients)
betas_enet = enet_model_fit.coef_
counter = 0
for col in boston.columns:
    if counter == 0:
        print("Beta weights/co-efficients - ELASTICNET")
        print("----------------------------------------")
    print(col + ": " + str(round(betas_enet[counter], 4)))
    counter +=1
```

```
from sklearn.linear_model import LinearRegression
lin_model = LinearRegression()

# fit the model to the training data
lin_model_fit = lin_model.fit(x, y)


# print the alpha value of the model (intercept)
print("Alpha/intercept (a)")
print(lin_model_fit.intercept_)

print("₩n")

# print the beta values of the model (co-efficients)
betas = lin_model_fit.coef_
counter = 0
for col in x.columns:
    if counter == 0:
        print("Beta weights/co-efficients (b1 to b13)")
        print("----------------------------------------")
    print(col + ": " + str(round(betas[counter], 4)))
    counter +=1
```

```
    counter += 1

        Alpha/intercept (a)
        0.4569495057182236


        Beta weights/co-efficients (b1 to b13)
        ----------------------------------------
        home: -0.0308
        education: 0.0008
        Number of Subjects: 0.0082
        device: 0.0484
        economic: 0.0874
        internet: 0.0071
        sports: 0.0059
        monitoring: -0.0291
        studytime: -0.0046
        sleeptime: -0.0091
        socialmedia: -0.002
        game: 0.0281
        groupstudies: 0.0052
        averagescored: -0.0702
        interaction: 0.0614
        dobuts: 0.0158
        interesting: -0.0095
        performance: 0.065


# predict every y value in the dataset
df_predict = lin_model_fit.predict(x)

# calculate RMSE (root mean square error) and R^2 (predictive power)
from sklearn.metrics import mean_squared_error, r2_score
rmse = (np.sqrt(mean_squared_error(y, df_predict)))
r2 = r2_score(y, df_predict)

# print the performance metrics
print("Model performance")
print("--------------------------------------")
print(f'RMSE is {rmse}')
print(f'R2 score is {r2}')

        Model performance
        --------------------------------------
        RMSE is 0.799198188661754
        R2 score is 0.07147785730066736


# SVM
from sklearn.svm import SVC
svm = SVC(C=100, kernel='rbf', gamma='auto')
svm.fit(x_train, y_train)

        SVC(C=100, gamma='auto')



accuracy = svm.score(x_test, y_test)
print('Accuracy: ', accuracy)
```

```
Accuracy:  0.5458937198067633
```

```
# KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)

# train the model using the training sets
knn.fit(x_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
y_pred_knn = knn.predict(x_test)
```

```
# Model Accuracy, how often is the classifier correct?
from sklearn import metrics
print("Accuracy : ", metrics.accuracy_score(y_test, y_pred_knn))
```

```
Accuracy :  0.5797101449275363
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
```

```
from sklearn.naive_bayes import GaussianNB
gauss=GaussianNB()
gauss.fit(x_train, y_train)
y_gauss=gauss.predict(x_test)
```

```
y_gauss
acc_gauss=round(accuracy_score(y_gauss, y_test)*100)
acc_gauss
```

```
58
```

```
f_gauss=f1_score(y_test,y_gauss)
f_gauss
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-57-fedde3272399> in <module>()
```

```
----> 1 f_gauss=f1_score(y_test,y_gauss)
      2 f_gauss
```

3 frames

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py in
_check_set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1365             raise ValueError(
```

```
r_gauss=roc_auc_score(y_test,y_gauss)
r_gauss
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-58-1f63fc01e7ea> in <module>()
----> 1 r_gauss=roc_auc_score(y_test,y_gauss)
      2 r_gauss

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_ranking.py in roc_auc_score(y_true, y_score,
average, sample_weight, max_fpr, multi_class, labels)
    558             )
    559         if multi_class == "raise":
--> 560             raise ValueError("multi_class must be in ('ovo', 'ovr')")
    561         return _multiclass_roc_auc_score(
    562             y_true, y_score, labels, multi_class, average, sample_weight

ValueError: multi_class must be in ('ovo', 'ovr')
```

SEARCH STACK OVERFLOW

```
from sklearn.linear_model import Perceptron
percep=Perceptron()
percep.fit(x_train, y_train)
y_percep=percep.predict(x_test)
y_percep
```

```
array([2, 0, 2, 2, 2, 2, 0, 0, 2, 1, 2, 2, 2, 2, 0, 2, 2, 0, 2, 0, 0, 0,
       2, 0, 0, 0, 2, 2, 0, 0, 1, 2, 0, 2, 1, 1, 2, 0, 2, 0, 2, 2, 0, 1,
       0, 0, 0, 2, 2, 2, 0, 2, 2, 0, 0, 2, 1, 1, 1, 2, 0, 0, 1, 2, 2, 1,
       2, 0, 2, 0, 2, 0, 2, 2, 2, 1, 1, 2, 2, 2, 0, 2, 2, 0, 0, 2, 0, 0,
       0, 0, 0, 1, 2, 1, 0, 2, 0, 0, 2, 0, 2, 1, 2, 0, 0, 1, 1, 0, 2, 2,
       0, 2, 1, 2, 0, 2, 0, 0, 0, 0, 2, 2, 1, 2, 1, 2, 2, 2, 1, 1, 1, 1,
       1, 0, 0, 2, 2, 2, 2, 2, 0, 2, 2, 2, 1, 2, 2, 0, 0, 0, 2, 2, 0, 1,
       0, 2, 2, 2, 0, 2, 2, 2, 2, 0, 0, 2, 2, 1, 2, 2, 2, 2, 0, 2, 2, 0,
       0, 2, 0, 2, 2, 1, 0, 2, 2, 0, 2, 0, 2, 0, 0, 2, 2, 0, 2, 0, 1, 2,
       2, 0, 0, 0, 1, 0, 2, 0, 2])
```

Decusuib Tree Classifier Model

```
from sklearn.tree import DecisionTreeClassifier
decitree=DecisionTreeClassifier()
decitree.fit(x_train, y_train)
y_decitree=decitree.predict(x_test)
y_decitree
```

```
array([0, 0, 2, 2, 0, 0, 0, 1, 0, 1, 0, 2, 0, 0, 1, 2, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 2, 0, 1, 1, 1, 2, 0, 0, 1, 1, 0, 2, 1, 0, 0, 2, 2, 0,
       0, 1, 0, 1, 2, 2, 1, 0, 0, 0, 2, 0, 0, 1, 0, 0, 1, 0, 1, 2, 2, 1,
```

```
0, 0, 2, 0, 2, 1, 2, 2, 0, 1, 1, 0, 0, 2, 1, 1, 2, 0, 0, 2, 2, 0,
1, 0, 0, 1, 2, 1, 0, 2, 0, 1, 0, 2, 1, 1, 2, 0, 1, 1, 2, 1, 2, 2,
1, 2, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 1, 2, 0, 2, 1, 1, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 2, 2, 2, 0, 1, 2, 0, 1, 0, 0, 0, 0, 0, 1,
0, 2, 0, 0, 0, 2, 2, 2, 2, 2, 0, 2, 2, 1, 2, 2, 2, 2, 0, 2, 0, 1,
0, 2, 0, 2, 0, 1, 0, 2, 0, 1, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
2, 0, 1, 2, 1, 1, 0, 1, 2])
```

```
acc_decitree=round(accuracy_score(y_decitree, y_test)*100)
acc_decitree
```

```
58
```

```
f_decitree=f1_score(y_test,y_decitree)
f_decitree
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-61-1fc41a587ec3> in <module>()
----> 1 f_decitree=f1_score(y_test,y_decitree)
      2 f_decitree

                                3 frames
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py in
_check_set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1365             raise ValueError(
   1366                 "Target is %s but average='binary'. Please "
-> 1367                  "choose another average setting, one of %r." % (y_type, average_options)
   1368             )
   1369     elif pos_label not in (None, 1):

ValueError: Target is multiclass but average='binary'. Please choose another average setting, one of
[None, 'micro', 'macro', 'weighted'].
```

```
r=decitree=roc_auc_score(y_test,y_decitree)
r_decitree
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
```

Random Forest Classfier Model

Random Forest Classifier Model

```
from sklearn.ensemble import RandomForestClassifier
random=RandomForestClassifier()
random.fit(x_train, y_train)
y_random=random.predict(x_test)
y_random
```

```
array([0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 2, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 2, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 2, 2, 1,
       2, 0, 0, 0, 0, 0, 0, 2, 0, 1, 1, 0, 0, 0, 1, 0, 2, 1, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 1, 1, 2, 0, 0, 1, 1, 0, 1,
       1, 1, 1, 2, 0, 0, 1, 0, 0, 0, 2, 2, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0,
       0, 2, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 2, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 1, 1, 1, 0, 1, 0])
```

```
acc_random=round(accuracy_score(y_random,y_test)*100)
acc_random
```

```
62
```

```
f_random=f1_score(y_test,y_random)
f_random
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-67-0cf64fa154a1> in <module>()
----> 1 f_random=f1_score(y_test,y_random)
      2 f_random

 3 frames
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py in
_check_set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1365             raise ValueError(
   1366                 "Target is %s but average='binary'. Please "
-> 1367                  "choose another average setting, one of %r." % (y_type, average_options)
   1368             )
   1369     elif pos_label not in (None, 1):

ValueError: Target is multiclass but average='binary'. Please choose another average setting, one of
[None, 'micro', 'macro', 'weighted'].
```

```
r_random=roc_auc_score(y_test,y_random)
r_random
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-68-e77253f5262f> in <module>()
```

```
----> 1 r_random=roc_auc_score(y_test,y_random)
      2 r_random

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_ranking.py in roc_auc_score(y_true, y_score,
average, sample_weight, max_fpr, multi_class, labels)
    558             )
    559         if multi_class == "raise":
--> 560             raise ValueError("multi_class must be in ('ovo', 'ovr')")
    561         return _multiclass_roc_auc_score(
    562             y_true, y_score, labels, multi_class, average, sample_weight
```

Gradient Boosting Classifier Model

SEARCH STACK OVERFLOW

```
from sklearn.ensemble import GradientBoostingClassifier
gbk=GradientBoostingClassifier()
gbk.fit(x_train, y_train)
y_gbk=gbk.predict(x_test)
y_gbk
```

```
array([0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 2, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 2, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 2, 2, 1,
       2, 0, 0, 0, 0, 0, 0, 2, 0, 1, 1, 0, 2, 0, 0, 2, 2, 1, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 2, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 2, 2, 0, 1, 0, 0, 0, 2, 2, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 1, 2, 2, 0, 2, 0, 0, 0, 2,
       0, 2, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 2, 0, 0, 1, 2, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 1, 1, 0, 0, 0])
```

```
acc_gbk=round(accuracy_score(y_gbk,y_test)*100)
acc_gbk
```

```
62
```

Choosing the GBK Model to predict the online satisfaction because the f1 score and accuracy score was high comparing with other models

✓ 0s completed at 18:50