```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
#visualizes all the columns
pd.set_option('display.max_columns',None)
```

```
#models
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

```
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn import metrics
from sklearn.metrics import mean_absolute_error,accuracy_score,precision_score,confusion_matrix,f1_score,classification_report
```

```
#Kepler Object=koi
```

```
data=pd.read_csv('/content/exoplanets_2018 (1).csv')
data
```

	kepid	kepoi_name	kepler_name	koi_disposition	koi_pdisposition	koi_score	koi_fpf
0	10797460	K00752.01	Kepler-227 b	CONFIRMED	CANDIDATE	1.000	
1	10797460	K00752.02	Kepler-227 c	CONFIRMED	CANDIDATE	0.969	
2	10811496	K00753.01	NaN	CANDIDATE	CANDIDATE	0.000	
3	10848459	K00754.01	NaN	FALSE POSITIVE	FALSE POSITIVE	0.000	
4	10854555	K00755.01	Kepler-664 b	CONFIRMED	CANDIDATE	1.000	
...	...	...	...	...	...	...	
9559	10090151	K07985.01	NaN	FALSE POSITIVE	FALSE POSITIVE	0.000	
9560	10128825	K07986.01	NaN	CANDIDATE	CANDIDATE	0.497	
9561	10147276	K07987.01	NaN	FALSE POSITIVE	FALSE POSITIVE	0.021	
9562	10155286	K07988.01	NaN	CANDIDATE	CANDIDATE	0.092	
9563	10156110	K07989.01	NaN	FALSE POSITIVE	FALSE POSITIVE	0.000	

9564 rows × 49 columns

Start coding or generate with AI.

```
data.isnull().sum()
```

```
kepid                  0
kepoi_name             0
kepler_name         7205
koi_disposition        0
koi_pdisposition       0
koi_score           1510
koi_fpflag_nt          0
koi_fpflag_ss          0
koi_fpflag_co          0
koi_fpflag_ec          0
koi_period             0
koi_period_err1      454
koi_period_err2      454
koi_time0bk            0
koi_time0bk_err1     454
koi_time0bk_err2     454
koi_impact           363
koi_impact_err1      454
koi_impact_err2      454
koi_duration           0
koi_duration_err1    454
koi_duration_err2    454
koi_depth            363
koi_depth_err1       454
koi_depth_err2       454
koi_prad             363
```

```
koi_prad_err1         363
koi_prad_err2         363
koi_teq               363
koi_teq_err1         9564
koi_teq_err2         9564
koi_insol             321
koi_insol_err1        321
koi_insol_err2        321
koi_model_snr         363
koi_tce_plnt_num      346
koi_tce_delivname     346
koi_steff             363
koi_steff_err1        468
koi_steff_err2        483
koi_slogg             363
koi_slogg_err1        468
koi_slogg_err2        468
koi_srad              363
koi_srad_err1         468
koi_srad_err2         468
ra                      0
dec                     0
koi_kepmag              1
dtype: int64
```

```
data = data.rename(columns={'kepid':'KepID',
# 'kepoi_name':'KOIName',
# 'kepler_name':'KeplerName',
# 'koi_disposition':'ExoplanetArchiveDisposition',
# 'koi_pdisposition':'DispositionUsingKeplerData',
'koi_score':'DispositionScore',
'koi_fpflag_nt':'NotTransit-LikeFalsePositiveFlag',
'koi_fpflag_ss':'koi_fpflag_ss',
'koi_fpflag_co':'CentroidOffsetFalsePositiveFlag',
'koi_fpflag_ec':'EphemerisMatchIndicatesContaminationFalsePositiveFlag',
'koi_period':'OrbitalPeriod[days',
'koi_period_err1':'OrbitalPeriodUpperUnc.[days',
'koi_period_err2':'OrbitalPeriodLowerUnc.[days',
'koi_time0bk':'TransitEpoch[BKJD',
'koi_time0bk_err1':'TransitEpochUpperUnc.[BKJD',
'koi_time0bk_err2':'TransitEpochLowerUnc.[BKJD',
'koi_impact':'ImpactParamete',
'koi_impact_err1':'ImpactParameterUpperUnc',
'koi_impact_err2':'ImpactParameterLowerUnc',
'koi_duration':'TransitDuration[hrs',
'koi_duration_err1':'TransitDurationUpperUnc.[hrs',
'koi_duration_err2':'TransitDurationLowerUnc.[hrs',
'koi_depth':'TransitDepth[ppm',
'koi_depth_err1':'TransitDepthUpperUnc.[ppm',
'koi_depth_err2':'TransitDepthLowerUnc.[ppm',
'koi_prad':'PlanetaryRadius[Earthradii',
'koi_prad_err1':'PlanetaryRadiusUpperUnc.[Earthradii',
'koi_prad_err2':'PlanetaryRadiusLowerUnc.[Earthradii',
'koi_teq':'EquilibriumTemperature[K',
# 'koi_teq_err1':'EquilibriumTemperatureUpperUnc.[K',
# 'koi_teq_err2':'EquilibriumTemperatureLowerUnc.[K',
'koi_insol':'InsolationFlux[Earthflux',
'koi_insol_err1':'InsolationFluxUpperUnc.[Earthflux',
'koi_insol_err2':'InsolationFluxLowerUnc.[Earthflux',
'koi_model_snr':'TransitSignal-to-Nois','koi_tce_plnt_num':'TCEPlanetNumbe',
'koi_tce_delivname':'TCEDeliver',
'koi_steff':'StellarEffectiveTemperature[K',
'koi_steff_err1':'StellarEffectiveTemperatureUpperUnc.[K',
'koi_steff_err2':'StellarEffectiveTemperatureLowerUnc.[K',
'koi_slogg':'StellarSurfaceGravity[log10(cm/s**2)',
'koi_slogg_err1':'StellarSurfaceGravityUpperUnc.[log10(cm/s**2)',
'koi_slogg_err2':'StellarSurfaceGravityLowerUnc.[log10(cm/s**2)',
'koi_srad':'StellarRadius[Solarradii',
'koi_srad_err1':'StellarRadiusUpperUnc.[Solarradii',
'koi_srad_err2':'StellarRadiusLowerUnc.[Solarradii',
'ra':'RA[decimaldegrees',
'dec':'Dec[decimaldegrees',
'koi_kepmag':'Kepler-band[mag]'
})
```

```
data.koi_disposition.value_counts()
```

```
koi_disposition
FALSE POSITIVE    4840
CANDIDATE         2367
```

```
CONFIRMED         2357
Name: count, dtype: int64
```

```
data.koi_pdisposition.value_counts()
```

```
koi_pdisposition
FALSE POSITIVE    4847
CANDIDATE         4717
Name: count, dtype: int64
```

Start coding or generate with AI.

```
from sklearn.preprocessing import LabelEncoder
lst=['koi_disposition','koi_pdisposition']
dict1={}
for col in lst:
  dict1[col]=LabelEncoder()
  data[col]=dict1[col].fit_transform(data[col])
dict1
data.head()
```

	KepID	kepoi_name	kepler_name	koi_disposition	koi_pdisposition	DispositionScore	Lik
0	10797460	K00752.01	Kepler-227 b	1	0	1.000	
1	10797460	K00752.02	Kepler-227 c	1	0	0.969	
2	10811496	K00753.01	NaN	0	0	0.000	
3	10848459	K00754.01	NaN	2	1	0.000	
4	10854555	K00755.01	Kepler-664 b	1	0	1.000	

```
data.drop(columns=['kepoi_name','kepler_name','koi_teq_err1','koi_teq_err2','TCEDeliver'],inplace=True)
```

```
data.head()
```

	KepID	koi_disposition	koi_pdisposition	DispositionScore	NotTransit-LikeFalsePositiveFlag	koi_f
0	10797460	1	0	1.000	0	
1	10797460	1	0	0.969	0	
2	10811496	0	0	0.000	0	
3	10848459	2	1	0.000	0	
4	10854555	1	0	1.000	0	

Double-click (or enter) to edit

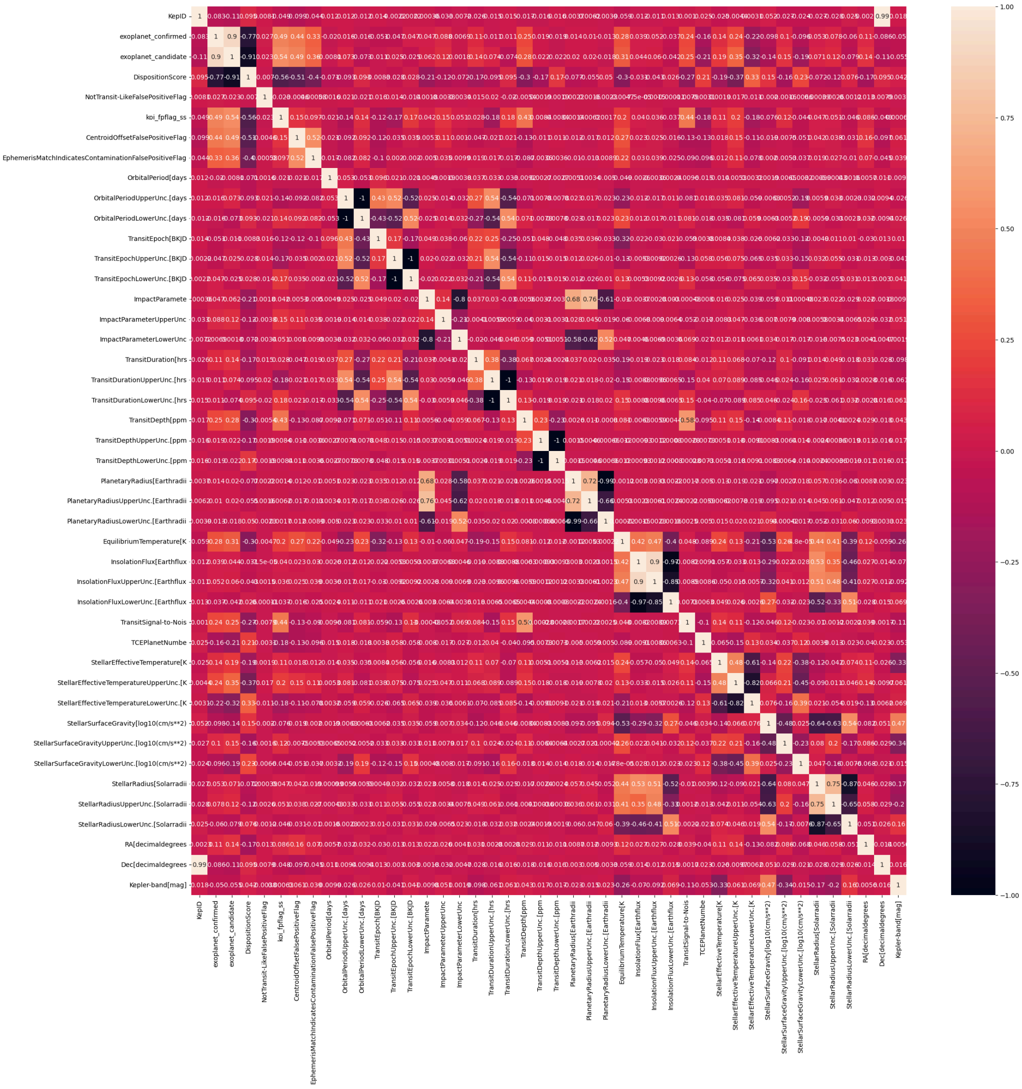
```
data = data.rename(columns={'koi_disposition':'exoplanet_confirmed','koi_pdisposition':'exoplanet_candidate'})
```

```
data.head()
```

	KepID	exoplanet_confirmed	exoplanet_candidate	DispositionScore	NotTransit-LikeFalsePositiveFlag
0	10797460	1	0	1.000	0
1	10797460	1	0	0.969	0
2	10811496	0	0	0.000	0
3	10848459	2	1	0.000	0
4	10854555	1	0	1.000	0

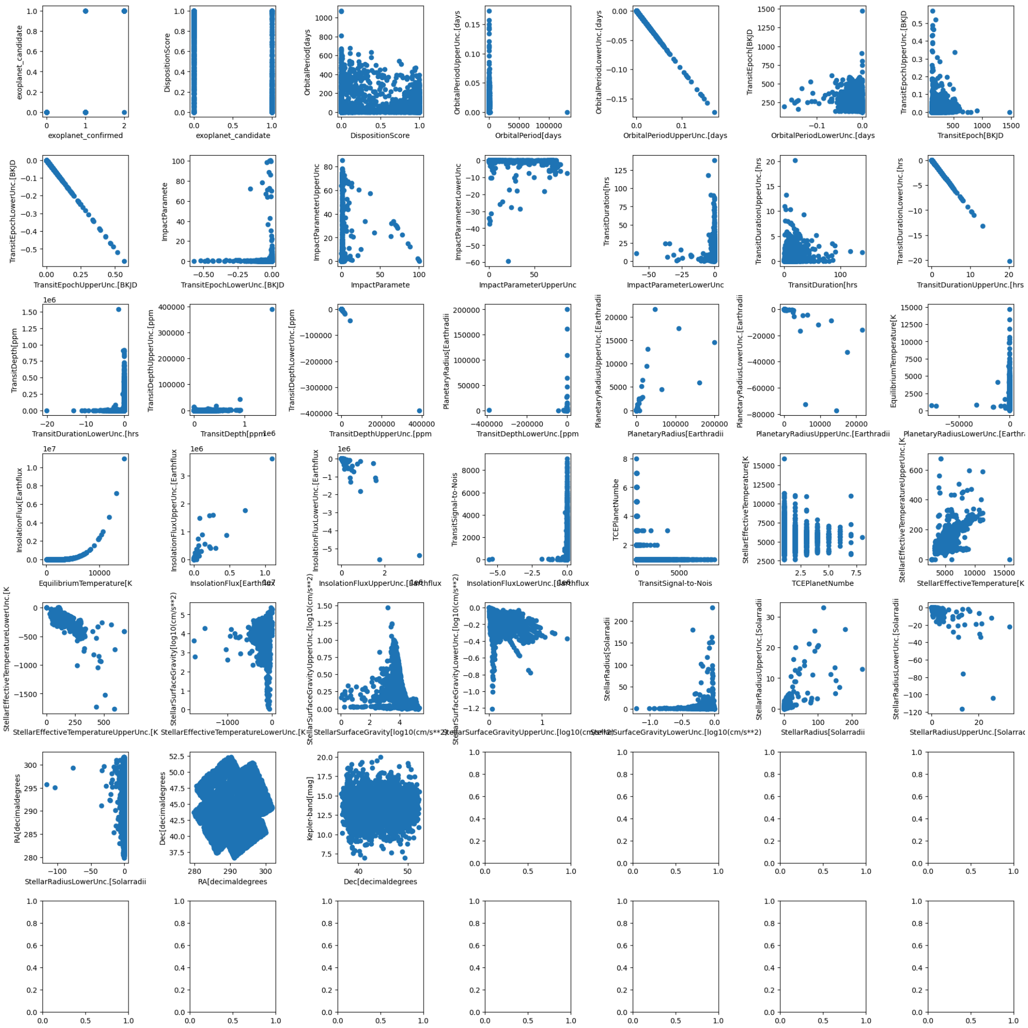
```
corelation = data.corr()#corelation between all the columns
fig, ax = plt.subplots(figsize=(25,25))#This line creates a new figure and a set of subplots #figsize(width and height)
sns.heatmap(corelation,xticklabels=corelation.columns,yticklabels=corelation.columns,annot=True,ax=ax)#we get corelation as a metrics .corel
```

```
<Axes: >
```



```
data.drop(columns=['KepID','NotTransit-LikeFalsePositiveFlag','koi_fpflag_ss','CentroidOffsetFalsePositiveFlag',
                   'EphemerisMatchIndicatesContaminationFalsePositiveFlag'], inplace=True)
```

```
# Create a list of column names
columns = data.columns
# Create subplots
fig, axes = plt.subplots(nrows=7, ncols=7, figsize=(20, 20))#create a grid of subplots using plt.subplots(nrows=7, ncols=7) to accommodate t
# Loop through each pair of columns and create scatter plots
for i,ax in enumerate(axes.flatten()):#loop through each pair of columns, create scatter plots for them, and assign them to individual subpl
    if i < len(columns) - 1:
        x_col = columns[i]
        y_col = columns[i + 1]
        ax.scatter(data[x_col], data[y_col])
        ax.set_xlabel(x_col)
        ax.set_ylabel(y_col)
# Adjust layout
plt.tight_layout()#Finally, we adjust the layout and display the plots using plt.tight_layout() and plt.show()
plt.show()
```



```
data
```

	exoplanet_confirmed	exoplanet_candidate	DispositionScore	OrbitalPeriod[days	OrbitalP
0	1	0	1.000	9.488036	
1	1	0	0.969	54.418383	
2	0	0	0.000	19.899140	
3	2	1	0.000	1.736952	
4	1	0	1.000	2.525592	
...	...	...	...	...	
9559	2	1	0.000	0.527699	
9560	0	0	0.497	1.739849	
9561	2	1	0.021	0.681402	
9562	0	0	0.092	333.486169	
9563	2	1	0.000	4.856035	

9564 rows × 39 columns

```
data.isnull().sum()
```

```
exoplanet_confirmed                                0
exoplanet_candidate                                0
DispositionScore                                1510
OrbitalPeriod[days                                 0
OrbitalPeriodUpperUnc.[days                      454
OrbitalPeriodLowerUnc.[days                      454
TransitEpoch[BKJD                                  0
TransitEpochUpperUnc.[BKJD                       454
TransitEpochLowerUnc.[BKJD                       454
ImpactParamete                                   363
ImpactParameterUpperUnc                          454
ImpactParameterLowerUnc                          454
TransitDuration[hrs                                0
TransitDurationUpperUnc.[hrs                     454
TransitDurationLowerUnc.[hrs                     454
TransitDepth[ppm                                 363
TransitDepthUpperUnc.[ppm                        454
TransitDepthLowerUnc.[ppm                        454
PlanetaryRadius[Earthradii                       363
PlanetaryRadiusUpperUnc.[Earthradii              363
PlanetaryRadiusLowerUnc.[Earthradii              363
EquilibriumTemperature[K                         363
InsolationFlux[Earthflux                         321
InsolationFluxUpperUnc.[Earthflux                321
InsolationFluxLowerUnc.[Earthflux                321
TransitSignal-to-Nois                            363
TCEPlanetNumbe                                   346
StellarEffectiveTemperature[K                    363
StellarEffectiveTemperatureUpperUnc.[K           468
StellarEffectiveTemperatureLowerUnc.[K           483
StellarSurfaceGravity[log10(cm/s**2)             363
StellarSurfaceGravityUpperUnc.[log10(cm/s**2)    468
StellarSurfaceGravityLowerUnc.[log10(cm/s**2)    468
StellarRadius[Solarradii                         363
StellarRadiusUpperUnc.[Solarradii                468
StellarRadiusLowerUnc.[Solarradii                468
RA[decimaldegrees                                  0
Dec[decimaldegrees                                 0
Kepler-band[mag]                                   1
dtype: int64
```

```
data.dropna(inplace=True)
```

```
# data.head()
```

```
data.shape
```

```
(7803, 39)
```

Start coding or generate with AI.

```
data['OrbitalPeriodUpperUnc.[days'].dtype
```

```
dtype('float64')
```

```
x=data.drop(columns='exoplanet_candidate')
```

```
data.exoplanet_candidate.value_counts()
```

```
exoplanet_candidate
0    4062
1    3741
Name: count, dtype: int64
```

```
y=data['exoplanet_candidate']
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4,random_state=1)
```

```
x_train.shape
```

```
(4681, 38)
```

```
x_test.shape
```

```
(3122, 38)
```

```
data.shape
```

```
(7803, 39)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train_scaled=sc.fit_transform(x_train)
```

```
x_test_scaled=sc.transform(x_test)
x_test_scaled
```

```
array([[ 0.93182479, -1.0348051 , -0.21374544, ...,  1.47444355,
        -0.9927248 , -0.33328488],
       [ 0.93182479, -1.0348051 , -0.2995078 , ...,  1.15858122,
        -0.92419694, -0.80284268],
       [ 0.93182479, -1.0348051 , -0.22834646, ...,  0.61429732,
         0.77317435,  1.37948309],
       ...,
       [-1.5931372 ,  1.02188583, -0.40101061, ..., -0.01312033,
         0.03086757,  0.05577729],
       [ 0.93182479, -1.0348051 ,  3.18527275, ...,  0.05171439,
        -1.39608929, -0.72234706],
       [-1.5931372 ,  0.43425985, -0.27984925, ...,  0.04694809,
        -0.33838837,  0.31440675]])
```

```
def evaluation(y_true, y_pred):

# Print Accuracy, Recall, F1 Score, and Precision metrics.
    print('Evaluation Metrics:')
    print('Accuracy: ' + str(metrics.accuracy_score(y_test, y_pred)))
    print('Recall: ' + str(metrics.recall_score(y_test, y_pred)))
    print('F1 Score: ' + str(metrics.f1_score(y_test, y_pred)))
    print('Precision: ' + str(metrics.precision_score(y_test, y_pred)))

# Print Confusion Matrix
    print('\nConfusion Matrix:')
    print(' TN,  FP, FN, TP')
    print(confusion_matrix(y_true, y_pred).ravel())

# Function Prints best parameters for GridSearchCV
def print_results(results):
    print('Best Parameters: {}\n'.format(results.best_params_))
```

```
lr=LogisticRegression(C=100, max_iter=200, class_weight='balanced')
# Fitting Model to the train set
lr.fit(x_train,y_train)

# Predicting on the test set
y_pred5=lr.predict(x_test)

# Evaluating model
evaluation(y_test, y_pred5)
```

```
Evaluation Metrics:
Accuracy: 0.8238308776425368
Recall: 0.7751004016064257
F1 Score: 0.8080949057920447
Precision: 0.8440233236151603

Confusion Matrix:
 TN,  FP, FN, TP
[1414  214  336 1158]
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

knn

```
knn=KNeighborsClassifier(leaf_size=8, metric='manhattan',weights='uniform')

# Fitting Model to the train set
knn.fit(x_train,y_train)

# Predicting on the test set
y_pred4=knn.predict(x_test)

# Evaluating model
evaluation(y_test, y_pred4)
```

```
Evaluation Metrics:
Accuracy: 0.8023702754644458
Recall: 0.7590361445783133
F1 Score: 0.7861351819757365
Precision: 0.8152408339324227

Confusion Matrix:
 TN,  FP, FN, TP
[1371  257  360 1134]
```

decision **tree**

```
tree=DecisionTreeClassifier()

# Fitting Model to the train set
tree.fit(x_train,y_train)

# Predicting on the test set
y_pred3=tree.predict(x_test)

# Evaluating model
evaluation(y_test,y_pred3)
```

```
Evaluation Metrics:
Accuracy: 0.9983984625240231
Recall: 0.998661311914324
F1 Score: 0.998327199732352
Precision: 0.9979933110367893

Confusion Matrix:
 TN,  FP, FN, TP
[1625    3    2 1492]
```

random **forest**

```
# Instantiate model
forest=RandomForestClassifier(n_estimators=100, criterion='gini')
# Fitting Model to the train set
forest.fit(x_train,y_train)
# Predicting on the test set
y_pred2=forest.predict(x_test)

# Evaluating model
evaluation(y_test, y_pred2)
```

```
Evaluation Metrics:
Accuracy: 0.9990390775144138
Recall: 0.998661311914324
F1 Score: 0.9989956478071643
Precision: 0.999330207635633

Confusion Matrix:
 TN,  FP, FN, TP
[1627    1    2 1492]
```

```
forest.feature_importances_
```

```
array([0.40626069, 0.25648283, 0.01474966, 0.00902676, 0.00910078,
       0.00147552, 0.0021883 , 0.00332403, 0.00750104, 0.00211219,
       0.00136105, 0.00417534, 0.00580974, 0.00926642, 0.01659487,
       0.00189965, 0.00180856, 0.05569416, 0.03553199, 0.04084757,
       0.01203039, 0.01711754, 0.01659179, 0.01017143, 0.00785109,
       0.00160457, 0.00172491, 0.01202405, 0.02231944, 0.00150562,
       0.00158019, 0.00148578, 0.00167941, 0.00142575, 0.00146524,
       0.00158551, 0.00097541, 0.00165069])
```

```
# Instantiate model
gr=GradientBoostingClassifier(n_estimators=1000,learning_rate=0.001,min_samples_split=10)
# Fitting Model to the train set
gr.fit(x_train,y_train)
# Predicting on the test set
y_pred1=gr.predict(x_test)
# Evaluating model
evaluation(y_test,y_pred1)
```

```
Evaluation Metrics:
Accuracy: 0.9990390775144138
Recall: 0.998661311914324
F1 Score: 0.9989956478071643
Precision: 0.999330207635633

Confusion Matrix:
 TN,  FP, FN, TP
[1627    1    2 1492]
```

```
x_im=gr.feature_importances_
x_im=x_im>10**-6
new=[]
for i,j in zip(x.columns,x_im):
  if j==True:
    new.append(i)
```

```
x_new=data[new]
```

```
dict10={'model':forest,'encoder':dict1[col]}
```

```
import pickle
with open('exo_planet_prediction.pkl','wb') as file:
    pickle.dump(dict10,file)
```

```
with open ('exo_planet_prediction.pkl','rb') as file1:
  var1=pickle.load(file1)
var1
```

```
{'model': RandomForestClassifier(), 'encoder': LabelEncoder()}
```

```
var1['model'].predict(x)
```

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