```
#Import Libraries
import numpy as np
import pandas as pd
import seaborn as sns
```

```
#Load the data
from google.colab import files # Use to load data on Google Colab
uploaded = files.upload() # Use to load data on Google Colab
```

```
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving WA_Fn-UseC_-HR-Employee-Attrition.csv to WA_Fn-UseC_-HR-Employee-Attrition.cs
```

```
#Store the data into the df variable
df = pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
df.head(7) #Print the first 7 rows
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educat
0	41	Yes	Travel_Rarely	1102	Sales	1	
1	49	No	Travel_Frequently	279	Research & Development	8	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	
4	27	No	Travel_Rarely	591	Research & Development	2	
5	32	No	Travel_Frequently	1005	Research & Development	2	
6	59	No	Travel_Rarely	1324	Research & Development	3	

```
#Get the number of rows and number of columns in the data
df.shape
```

```
(1470, 35)
```

```
#Get the column data types
df.dtypes
```

```
Age                        int64
Attrition                 object
BusinessTravel            object
DailyRate                  int64
```

```
Department                 object
DistanceFromHome            int64
Education                   int64
EducationField             object
EmployeeCount               int64
EmployeeNumber              int64
EnvironmentSatisfaction     int64
Gender                     object
HourlyRate                  int64
JobInvolvement              int64
JobLevel                    int64
JobRole                    object
JobSatisfaction             int64
MaritalStatus              object
MonthlyIncome               int64
MonthlyRate                 int64
NumCompaniesWorked          int64
Over18                     object
OverTime                   object
PercentSalaryHike           int64
PerformanceRating           int64
RelationshipSatisfaction    int64
StandardHours               int64
StockOptionLevel            int64
TotalWorkingYears           int64
TrainingTimesLastYear       int64
WorkLifeBalance             int64
YearsAtCompany              int64
YearsInCurrentRole          int64
YearsSinceLastPromotion     int64
YearsWithCurrManager        int64
dtype: object
```

```
#Count the empty (NaN, NAN, na) values in each column
df.isna().sum()
```

```
Age                        0
Attrition                  0
BusinessTravel             0
DailyRate                  0
Department                 0
DistanceFromHome           0
Education                  0
EducationField             0
EmployeeCount              0
EmployeeNumber             0
EnvironmentSatisfaction    0
Gender                     0
HourlyRate                 0
JobInvolvement             0
JobLevel                   0
JobRole                    0
JobSatisfaction            0
MaritalStatus              0
MonthlyIncome              0
MonthlyRate                0
NumCompaniesWorked         0
Over18                     0
OverTime                   0
```

```
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

```
#Another check for any null / missing values
df.isnull().values.any()
```

```
False
```

```
#View some basic statistical details like percentile, mean, standard deviation etc.
df.describe()
```

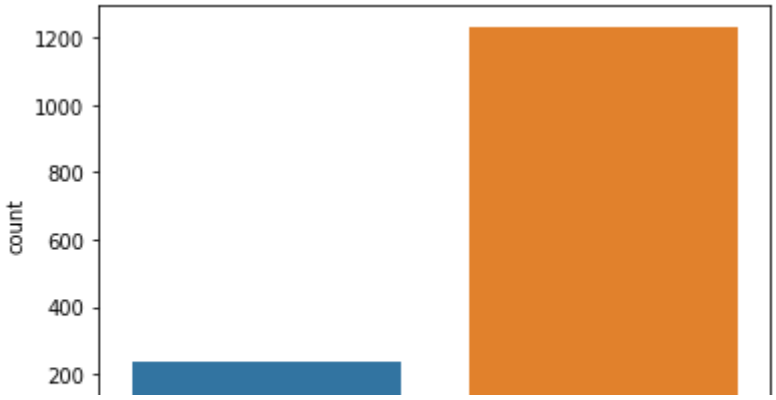
	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	Employ
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	147
mean	36.923810	802.485714	9.192517	2.912925	1.0	102
std	9.135373	403.509100	8.106864	1.024165	0.0	60
min	18.000000	102.000000	1.000000	1.000000	1.0	
25%	30.000000	465.000000	2.000000	2.000000	1.0	49
50%	36.000000	802.000000	7.000000	3.000000	1.0	102
75%	43.000000	1157.000000	14.000000	4.000000	1.0	155
max	60.000000	1499.000000	29.000000	5.000000	1.0	206

```
#Get a count of the number of employee attrition, the number of employees that stayed (no)
df['Attrition'].value_counts()
```

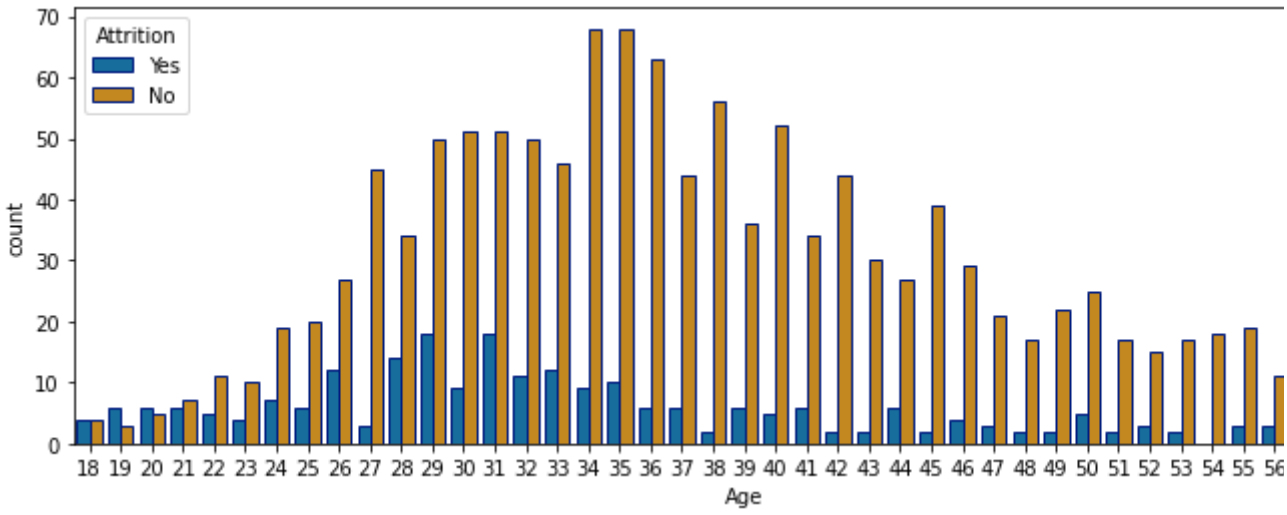
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

```
#Visualize this count
sns.countplot(df['Attrition'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4efd5a4150>
```




```
#Show the number of employees that left and stayed by age
import matplotlib.pyplot as plt
fig_dims = (12, 4)
fig, ax = plt.subplots(figsize=fig_dims)

#ax = axis
sns.countplot(x='Age', hue='Attrition', data = df, palette="colorblind", ax = ax,  edgecol
```




```
#Print all of the object data types and their unique values
for column in df.columns:
    if df[column].dtype == object:
        print(str(column) + ' : ' + str(df[column].unique()))
        print(df[column].value_counts())
        print("_____________________________________________________________________")
```

```
Attrition : ['Yes' 'No']
No     1233
Yes     237
Name: Attrition, dtype: int64
_____________________________________________________________________
BusinessTravel : ['Travel_Rarely' 'Travel_Frequently' 'Non-Travel']
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64
_____________________________________________________________________
```

```
Department : ['Sales' 'Research & Development' 'Human Resources']
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
_____________________________________________________________________
EducationField : ['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical Degree
 'Human Resources']
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
_____________________________________________________________________
Gender : ['Female' 'Male']
Male      882
Female    588
Name: Gender, dtype: int64
_____________________________________________________________________
JobRole : ['Sales Executive' 'Research Scientist' 'Laboratory Technician'
 'Manufacturing Director' 'Healthcare Representative' 'Manager'
 'Sales Representative' 'Research Director' 'Human Resources']
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64
_____________________________________________________________________
MaritalStatus : ['Single' 'Married' 'Divorced']
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
_____________________________________________________________________
Over18 : ['Y']
Y    1470
Name: Over18, dtype: int64
_____________________________________________________________________
OverTime : ['Yes' 'No']
No     1054
```

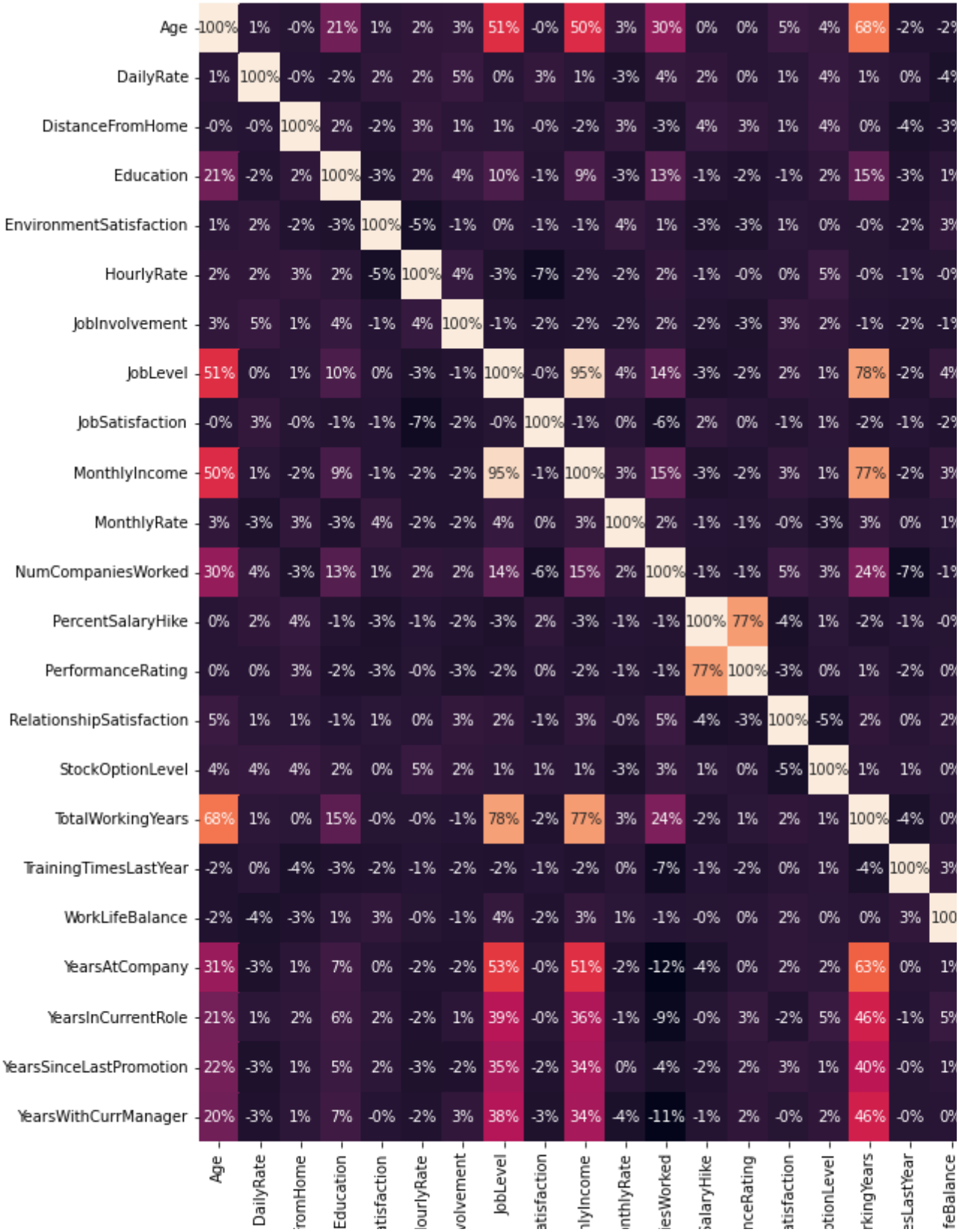
```
#Remove unneeded columns

#Remove the column EmployeeNumber
df = df.drop('EmployeeNumber', axis = 1) # A number assignment
#Remove the column StandardHours
df = df.drop('StandardHours', axis = 1) #Contains only value 80
#Remove the column EmployeeCount
df = df.drop('EmployeeCount', axis = 1) #Contains only the value 1
#Remove the column EmployeeCount
df = df.drop('Over18', axis = 1) #Contains only the value 'Yes'
```

```
#Get the correlation of the columns
df.corr()
```

	Age	DailyRate	DistanceFromHome	Education	Environme
Age	1.000000	0.010661	-0.001686	0.208034	
DailyRate	0.010661	1.000000	-0.004985	-0.016806	
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	
Education	0.208034	-0.016806	0.021042	1.000000	
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	
HourlyRate	0.024287	0.023381	0.031131	0.016775	
JobInvolvement	0.029820	0.046135	0.008783	0.042438	
JobLevel	0.509604	0.002966	0.005303	0.101589	
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	

```
#Visualize the correlation
plt.figure(figsize=(14,14))  #14in by 14in
sns.heatmap(df.corr(), annot=True, fmt='.0%')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4efce37910>
```




```
#Transform non-numeric columns into numerical columns
from sklearn.preprocessing import LabelEncoder

for column in df.columns:
        if df[column].dtype == np.number:
            continue
        df[column] = LabelEncoder().fit_transform(df[column])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: DeprecationWarning:
```

```
"""
```

```
#Create a new column at the end of the dataframe that contains the same value
df['Age_Years'] = df['Age']
#Remove the first column called age
df = df.drop('Age', axis = 1)
#Show the dataframe
df
```

	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	1	2	624	2	0	1
1	0	1	113	1	7	0
2	1	2	805	1	1	1
3	0	1	820	1	2	3
4	0	2	312	1	1	0
...	...	...	...	...	...	...
1465	0	1	494	1	22	1
1466	0	2	327	1	5	0
1467	0	2	39	1	3	2
1468	0	1	579	2	1	2
1469	0	2	336	1	7	2

1470 rows × 31 columns

```
#Split the data into independent 'X' and dependent 'Y' variables
X = df.iloc[:, 1:df.shape[1]].values
Y = df.iloc[:, 0].values
```

```
# Split the dataset into 75% Training set and 25% Testing set
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state =
```

```
#Use Random Forest Classification algorithm
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0
forest.fit(X_train, Y_train)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

```
#Get the accuracy on the training data
forest.score(X_train, Y_train)
```

```
     0.9791288566243194
```

```
#Show the confusion matrix and accuracy for  the model on the test data
#Classification accuracy is the ratio of correct predictions to total predictions made.
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(Y_test, forest.predict(X_test))

TN = cm[0][0]
TP = cm[1][1]
FN = cm[1][0]
FP = cm[0][1]

print(cm)
print('Model Testing Accuracy = "{}!"'.format(  (TP + TN) / (TP + TN + FN + FP)))
print()# Print a new line
```

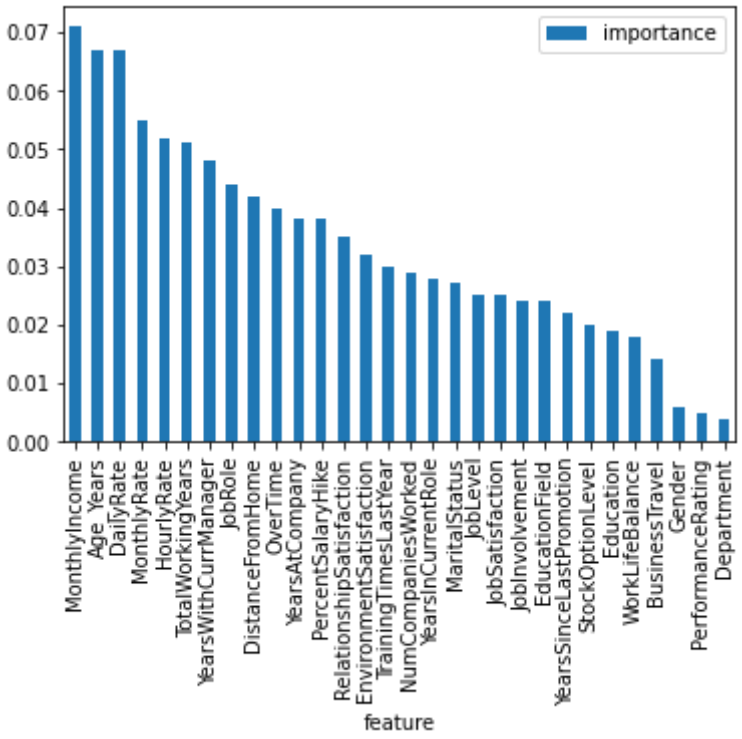
```
     [[309   1]
      [ 49   9]]
     Model Testing Accuracy = "0.8641304347826086!"
```

```
# Return the feature importances (the higher, the more important the feature).
importances = pd.DataFrame({'feature':df.iloc[:, 1:df.shape[1]].columns,'importance':np.ro
importances = importances.sort_values('importance',ascending=False).set_index('feature')
importances
```

feature	importance
MonthlyIncome	0.071
Age_Years	0.067
DailyRate	0.067
MonthlyRate	0.055
HourlyRate	0.052
TotalWorkingYears	0.051
YearsWithCurrManager	0.048
JobRole	0.044
DistanceFromHome	0.042
OverTime	0.040
YearsAtCompany	0.038
PercentSalaryHike	0.038
RelationshipSatisfaction	0.035
EnvironmentSatisfaction	0.032

```
#Visualize the importance
importances.plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ef44e8390>
```



Department	0.004

```
import numpy as np
```

```
input = np.array([   0,  341,    2,    9,    3,    3,    1,    1,    2,    2,    2,
          7,    3,    2, 1067,  759,    1,    0,    3,    0,    3,    0,
         10,    3,    1,   10,    3,    9,    7,   18])

new_input = input.reshape(1,-1)

prediction = forest.predict(new_input)

for x in prediction:
  if x == 1:
    print("yes")
  else:
    print("no")
```

```
     no
```

```
import numpy as np

input = np.array([  1, 221,   1,  23,   1,   1,   3,   1,  43,   1,   0,   6,   3,
         2, 177, 730,   1,   1,  13,   1,   1,   0,   1,   3,   1,   1,
         0,   1,   0,  11])

new_input = input.reshape(1,-1)

prediction = forest.predict(new_input)

for x in prediction:
  if x == 1:
    print("yes")
  else:
    print("no")
```

```
     yes
```