IMPORTING LIBRARIES

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

from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import  LabelEncoder,MinMaxScaler


from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB

from sklearn import metrics
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix


data=pd.read_csv('/content/survey.csv')


data.shape
```

```
(1259, 27)
```

```
data.describe()
```

	Age
count	1.259000e+03
mean	7.942815e+07
std	2.818299e+09
min	-1.726000e+03
25%	2.700000e+01
50%	3.100000e+01
75%	3.600000e+01
max	1.000000e+11

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Timestamp                  1259 non-null   object
 1   Age                        1259 non-null   int64 
 2   Gender                     1259 non-null   object
 3   Country                    1259 non-null   object
 4   state                      744 non-null    object
 5   self_employed              1241 non-null   object
 6   family_history             1259 non-null   object
 7   treatment                  1259 non-null   object
 8   work_interfere             995 non-null    object
 9   no_employees               1259 non-null   object
 10  remote_work                1259 non-null   object
 11  tech_company               1259 non-null   object
 12  benefits                   1259 non-null   object
 13  care_options               1259 non-null   object
 14  wellness_program           1259 non-null   object
 15  seek_help                  1259 non-null   object
 16  anonymity                  1259 non-null   object
 17  leave                      1259 non-null   object
 18  mental_health_consequence  1259 non-null   object
 19  phys_health_consequence    1259 non-null   object
 20  coworkers                  1259 non-null   object
 21  supervisor                 1259 non-null   object
 22  mental_health_interview    1259 non-null   object
 23  phys_health_interview      1259 non-null   object
 24  mental_vs_physical         1259 non-null   object
 25  obs_consequence            1259 non-null   object
 26  comments                   164 non-null    object
dtypes: int64(1), object(26)
memory usage: 265.7+ KB
```

```
data.head()
```

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	wor
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes	
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No	
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No	

5 rows × 27 columns

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

	0
Timestamp	0
Age	0
Gender	0
Country	0
state	515
self_employed	18
family_history	0
treatment	0
work_interfere	264
no_employees	0
remote_work	0
tech_company	0
benefits	0
care_options	0
wellness_program	0
seek_help	0
anonymity	0
leave	0
mental_health_consequence	0
phys_health_consequence	0
coworkers	0
supervisor	0
mental_health_interview	0
phys_health_interview	0
mental_vs_physical	0
obs_consequence	0
comments	1095

dtype: int64

```
data.drop(['comments'], axis= 1, inplace=True)
data.drop(['state'], axis= 1, inplace=True)
data.drop(['Timestamp'], axis= 1, inplace=True)


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

	0
Age	0
Gender	0
Country	0
self_employed	18
family_history	0
treatment	0
work_interfere	264
no_employees	0
remote_work	0
tech_company	0
benefits	0
care_options	0
wellness_program	0
seek_help	0
anonymity	0
leave	0
mental_health_consequence	0
phys_health_consequence	0
coworkers	0
supervisor	0
mental_health_interview	0
phys_health_interview	0
mental_vs_physical	0
obs_consequence	0

dtype: int64

```
    defaultInt = 0
    defaultString = 'NaN'
    defaultFloat = 0.0


    intFeatures = ['Age']
    stringFeatures = ['Gender', 'Country', 'self_employed', 'family_history', 'treatment', 'work
                      'no_employees', 'remote_work', 'tech_company', 'anonymity', 'leave', 'menta
                      'phys_health_consequence', 'coworkers', 'supervisor', 'mental_health_interv
                      'mental_vs_physical', 'obs_consequence', 'benefits', 'care_options', 'wellr
                      'seek_help']
    floatFeatures = []


    for feature in data:
        if feature in intFeatures:
            data[feature] = data[feature].fillna(defaultInt)
        elif feature in stringFeatures:
            data[feature] = data[feature].fillna(defaultString)
        elif feature in floatFeatures:
            data[feature] = data[feature].fillna(defaultFloat)
        else:
            print('Error: Feature %s not recognized.' % feature)


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

	0
Age	0
Gender	0
Country	0
self_employed	0
family_history	0
treatment	0
work_interfere	0
no_employees	0
remote_work	0
tech_company	0
benefits	0
care_options	0
wellness_program	0
seek_help	0
anonymity	0
leave	0
mental_health_consequence	0
phys_health_consequence	0
coworkers	0
supervisor	0
mental_health_interview	0
phys_health_interview	0
mental_vs_physical	0
obs_consequence	0

dtype: int64

```
data = data.drop(['Country'], axis= 1)
data.head()
```

	Age	Gender	self_employed	family_history	treatment	work_interfere	no_employees
0	37	Female	NaN	No	Yes	Often	6-25
1	44	M	NaN	No	No	Rarely	More than 1000
2	32	Male	NaN	No	No	Rarely	6-25
3	31	Male	NaN	Yes	Yes	Often	26-100
4	31	Male	NaN	No	No	Never	100-500

5 rows × 23 columns

```
gender = data['Gender'].unique()
print(gender)
```

```
['Female' 'M' 'Male' 'male' 'female' 'm' 'Male-ish' 'maile' 'Trans-female'
 'Cis Female' 'F' 'something kinda male?' 'Cis Male' 'Woman' 'f' 'Mal'
 'Male (CIS)' 'queer/she/they' 'non-binary' 'Femake' 'woman' 'Make' 'Nah'
 'All' 'Enby' 'fluid' 'Genderqueer' 'Female ' 'Androgyne' 'Agender'
 'cis-female/femme' 'Guy (-ish) ^_^' 'male leaning androgynous' 'Male '
 'Man' 'Trans woman' 'msle' 'Neuter' 'Female (trans)' 'queer'
 'Female (cis)' 'Mail' 'cis male' 'A little about you' 'Malr' 'p' 'femail'
 'Cis Man' 'ostensibly male, unsure what that really means']
```

```
male_str = ["male", "m", "male-ish", "maile", "mal", "male (cis)", "make", "male ", "man","m
trans_str = ["trans-female", "something kinda male?", "queer/she/they", "non-binary","nah",
female_str = ["cis female", "f", "female", "woman",  "femake", "female ","cis-female/femme",

for (row, col) in data.iterrows():

    if str.lower(col.Gender) in male_str:
        data['Gender'].replace(to_replace=col.Gender, value='male', inplace=True)

    if str.lower(col.Gender) in female_str:
        data['Gender'].replace(to_replace=col.Gender, value='female', inplace=True)

    if str.lower(col.Gender) in trans_str:
        data['Gender'].replace(to_replace=col.Gender, value='trans', inplace=True)

#Get rid of bullshit
stk_list = ['A little about you', 'p']
data = data[~data['Gender'].isin(stk_list)]

print(data['Gender'].unique())
```

```
<ipython-input-14-b168a9b6cc25>:11: FutureWarning: A value is trying to be set on a copy
The behavior will change in pandas 3.0. This inplace method will never work because the

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col


  data['Gender'].replace(to_replace=col.Gender, value='female', inplace=True)
<ipython-input-14-b168a9b6cc25>:8: FutureWarning: A value is trying to be set on a copy
The behavior will change in pandas 3.0. This inplace method will never work because the

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col


  data['Gender'].replace(to_replace=col.Gender, value='male', inplace=True)
<ipython-input-14-b168a9b6cc25>:14: FutureWarning: A value is trying to be set on a copy
The behavior will change in pandas 3.0. This inplace method will never work because the

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col


  data['Gender'].replace(to_replace=col.Gender, value='trans', inplace=True)
['female' 'male' 'trans']
```

```
data['self_employed'] =data['self_employed'].replace([defaultString], 'No')
print(data['self_employed'].unique())
```

```
['No' 'Yes']
```

```
data['work_interfere'] = data['work_interfere'].replace([defaultString], 'Don\'t know' )
print(data['work_interfere'].unique())
```

```
['Often' 'Rarely' 'Never' 'Sometimes' "Don't know"]
```

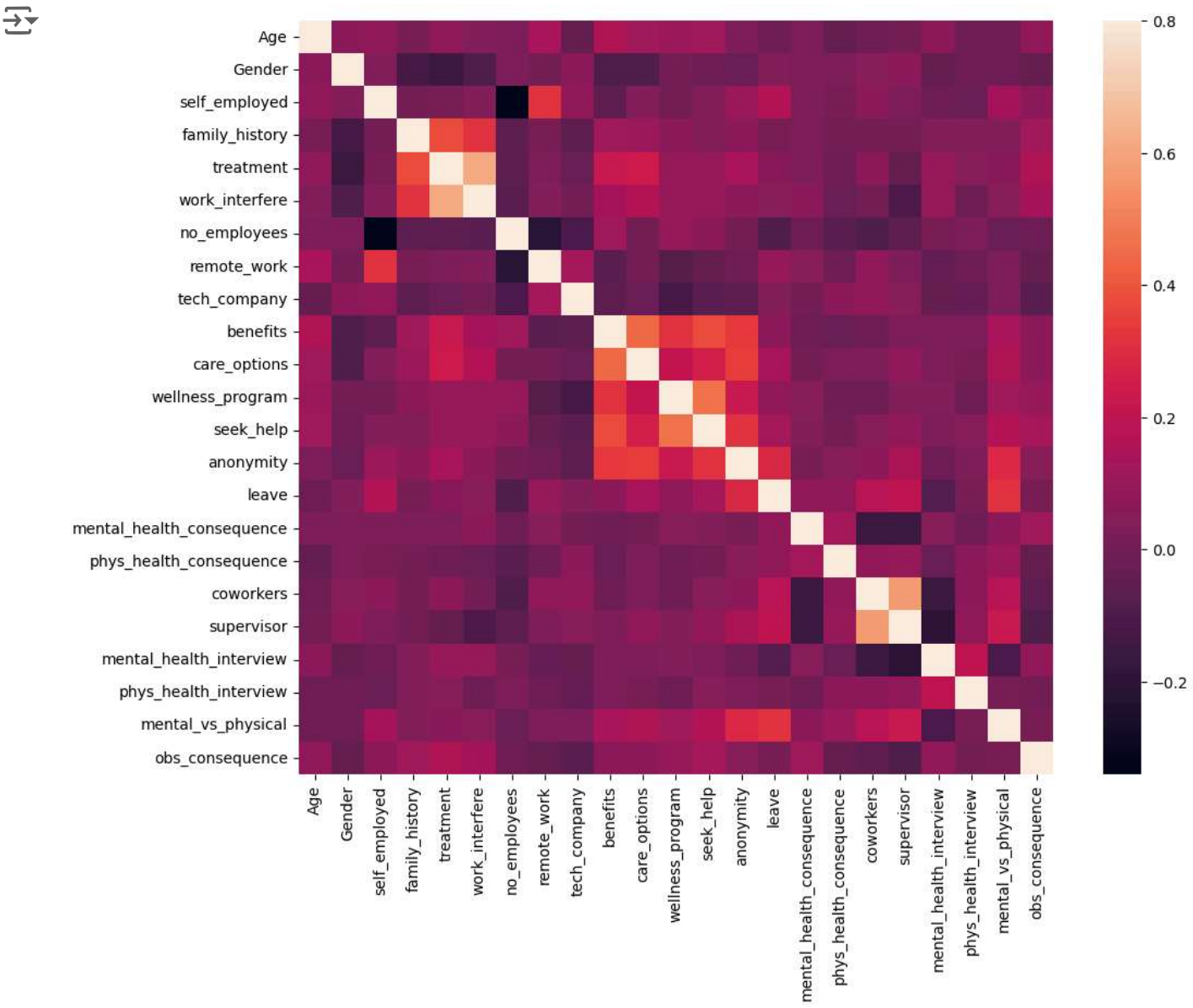
```
labelDict = {}
for feature in data:
    le = preprocessing.LabelEncoder()
    le.fit(data[feature])
    le_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
    data[feature] = le.transform(data[feature])
    labelKey = 'label_' + feature
    labelValue = [*le_name_mapping]
    labelDict[labelKey] =labelValue

for key, value in labelDict.items():
    print(key, value)
```

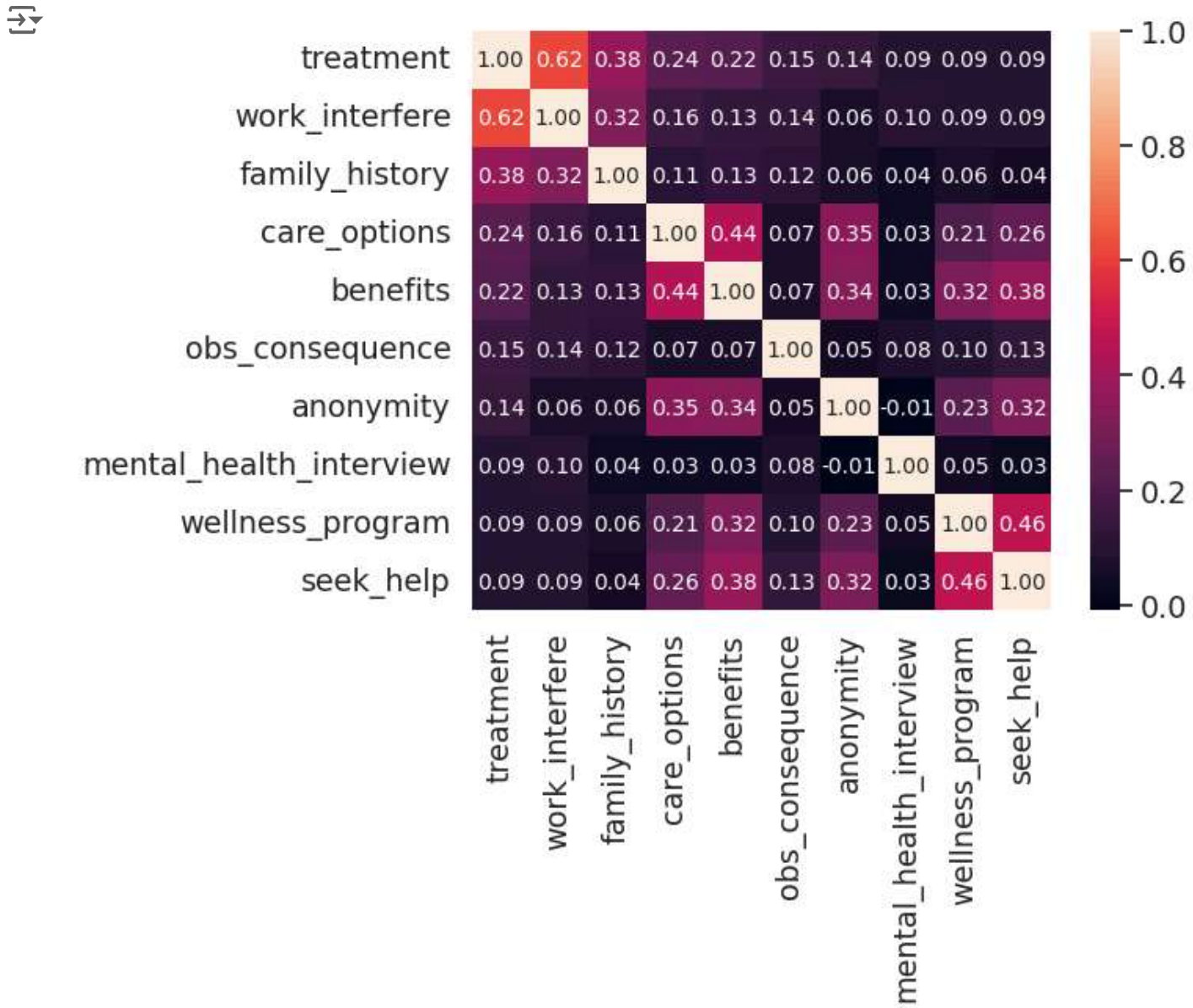
```
label_Age [-1726, -29, 5, 11, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32
label_Gender ['female', 'male', 'trans']
label_self_employed ['No', 'Yes']
label_family_history ['No', 'Yes']
```

```
        label_treatment ['No', 'Yes']
        label_work_interfere ["Don't know", 'Never', 'Often', 'Rarely', 'Sometimes']
        label_no_employees ['1-5', '100-500', '26-100', '500-1000', '6-25', 'More than 1000']
        label_remote_work ['No', 'Yes']
        label_tech_company ['No', 'Yes']
        label_benefits ["Don't know", 'No', 'Yes']
        label_care_options ['No', 'Not sure', 'Yes']
        label_wellness_program ["Don't know", 'No', 'Yes']
        label_seek_help ["Don't know", 'No', 'Yes']
        label_anonymity ["Don't know", 'No', 'Yes']
        label_leave ["Don't know", 'Somewhat difficult', 'Somewhat easy', 'Very difficult', 'Ver
        label_mental_health_consequence ['Maybe', 'No', 'Yes']
        label_phys_health_consequence ['Maybe', 'No', 'Yes']
        label_coworkers ['No', 'Some of them', 'Yes']
        label_supervisor ['No', 'Some of them', 'Yes']
        label_mental_health_interview ['Maybe', 'No', 'Yes']
        label_phys_health_interview ['Maybe', 'No', 'Yes']
        label_mental_vs_physical ["Don't know", 'No', 'Yes']
        label_obs_consequence ['No', 'Yes']
```

```
corrmat = data.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
plt.show()
```


Age
Gender
self_employed
family_history
treatment
work_interfere
no_employees
remote_work
tech_company
benefits
care_options
wellness_program
seek_help
anonymity
leave
mental_health_consequence
phys_health_consequence
coworkers
supervisor
mental_health_interview
phys_health_interview
mental_vs_physical
obs_consequence
0.8
0.6
0.4
0.2
0.0
−0.2

```
k=10
cols = corrmat.nlargest(k, 'treatment')['treatment'].index
cm = np.corrcoef(data[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 10},
plt.show()
```




```
scaler = MinMaxScaler()
data['Age'] = scaler.fit_transform(data[['Age']])
data.head()
```

	Age	Gender	self_employed	family_history	treatment	work_interfere	no_employees
0	0.46	0	0	0	1	2	4
1	0.60	1	0	0	0	3	5
2	0.36	1	0	0	0	3	4
3	0.34	1	0	1	1	2	2
4	0.34	1	0	0	0	1	1

5 rows × 23 columns

```
feature_cols = ['Age', 'Gender', 'family_history', 'benefits', 'care_options', 'anonymity',
X = data[feature_cols]
y = data.treatment
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)


models = {
    "Logistic Regression":LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Naive Bayes": GaussianNB()
}

# Evaluate each model
for name, model in models.items():
    print(f"--- {name} ---")
    # Train the model
    model.fit(X_train, y_train)
    # Make predictions
    y_pred = model.predict(X_test)
    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    # Display results
    print(f"Accuracy: {accuracy:.2f}")
    print("\nConfusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))
    print("\n")
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.75      0.75       191
           1       0.75      0.75      0.75       187

    accuracy                           0.75       378
   macro avg       0.75      0.75      0.75       378
weighted avg       0.75      0.75      0.75       378



--- Random Forest ---
Accuracy: 0.80

Confusion Matrix:
[[144  47]
 [ 30 157]]

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.75      0.79       191
           1       0.77      0.84      0.80       187

    accuracy                           0.80       378
   macro avg       0.80      0.80      0.80       378
weighted avg       0.80      0.80      0.80       378
```