Support Vector Machine Algorithm (SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

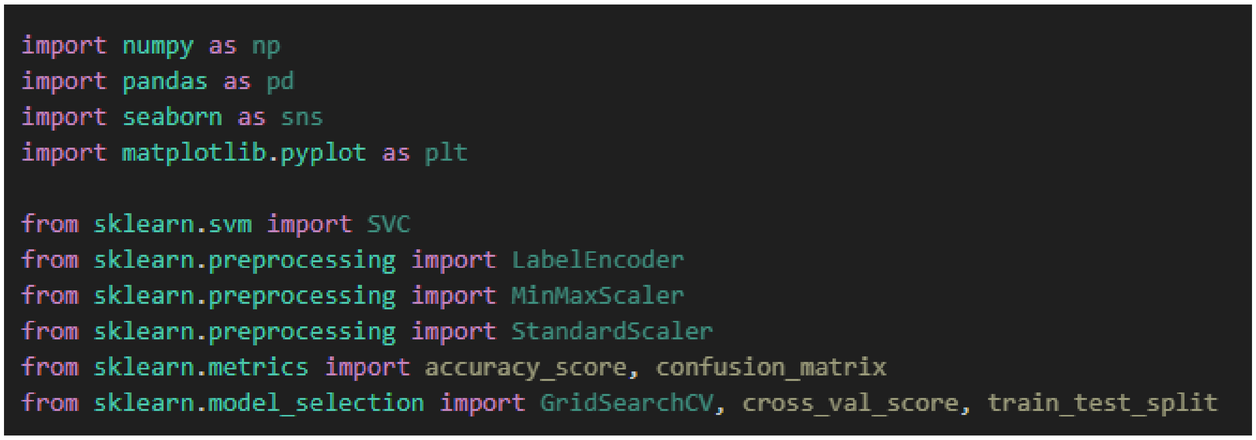
The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine.

There are two different types of SVMs, each used for different things :
• Simple SVM : Typically used for linear regression and classification problems.
• Kernel SVM : Has more flexibility for non-linear data because you can add more features to fit a hyperplane instead of a two-dimensional space.

This assignment will study following Questions :
Problem Statement No 1 :
Classify the Size Categorie using SVM (forestfires.csv) - the burned area of the forest (Small, Large)

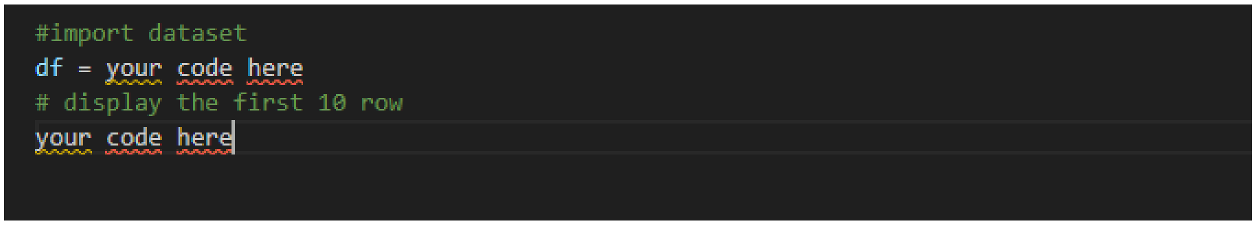
1. Import necessary libraries (you should be able to install any libraries that have not been installed in your environment so far)



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

from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import GridSearchCV, cross_val_score, train_test_split
```

2. Import dataset (read data to a pandas dataframe “df” from the “forestfires.csv”, and display the first 10 row of the dataset)

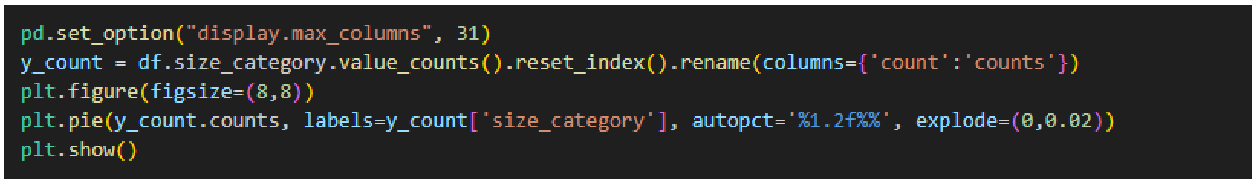


```
#import dataset
df = your code here
# display the first 10 row
your code here
```

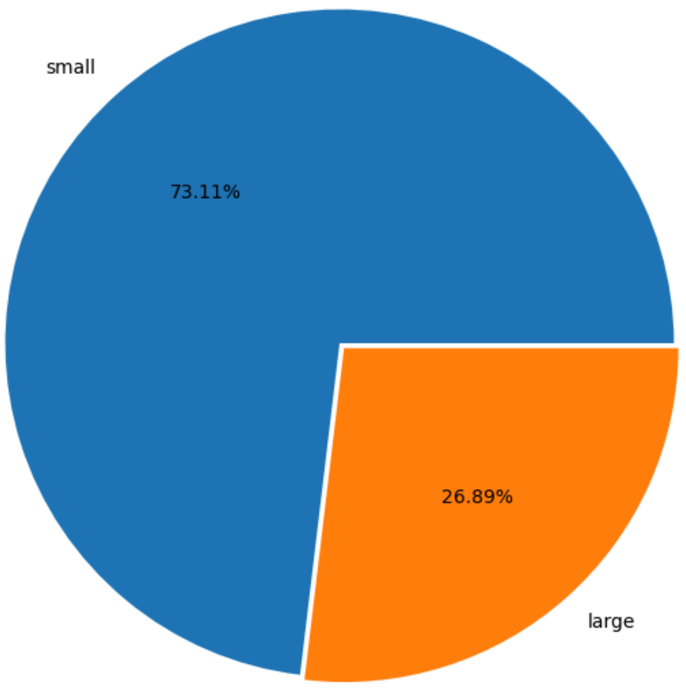
3. Data understanding (use dataframe attributes, methods, and other python methods to investigate the dataset. Answer the below questions)
    a. What is the dataset size
    b. Give the data description of the below column with the the column meaning and the value range
    Examples:
        - month: month of the year: 'jan' to 'dec'
        - FFMC: FFMC index from the FWI system: 18.7 to 96.20
        - temp: temperature in Celsius degrees: 2.2 to 33.30

    Data description for the below column
        - day:
        - DMC:
        - DC:
        - ISI:
        - RH:
        - wind:
        - rain:
        - Size_Categorie:
4. Data visualization

    Example 1: pie figure

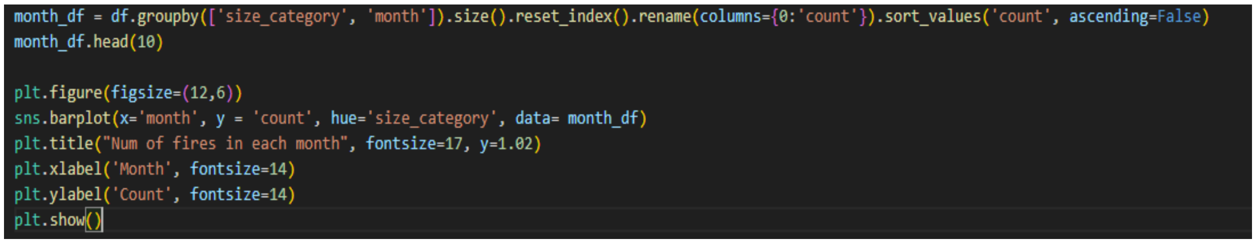


```
pd.set_option("display.max_columns", 31)
y_count = df.size_category.value_counts().reset_index().rename(columns={'count':'counts'})
plt.figure(figsize=(8,8))
plt.pie(y_count.counts, labels=y_count['size_category'], autopct='%1.2f%%', explode=(0,0.02))
plt.show()
```



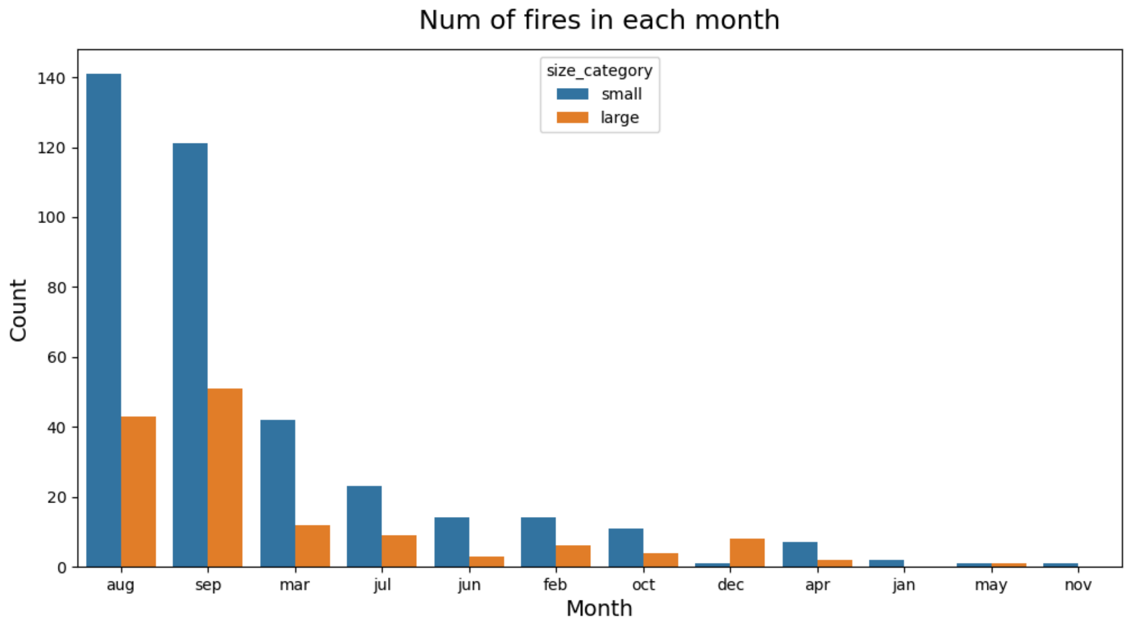
Take example 1 as a reference, plot the pie figure of the number of fires in August

Example 2: number of fires in each month



```
month_df = df.groupby(['size_category', 'month']).size().reset_index().rename(columns={0:'count'}).sort_values('count', ascending=False)
month_df.head(10)

plt.figure(figsize=(12,6))
sns.barplot(x='month', y = 'count', hue='size_category', data= month_df)
plt.title("Num of fires in each month", fontsize=17, y=1.02)
plt.xlabel('Month', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.show()
```



Observation:

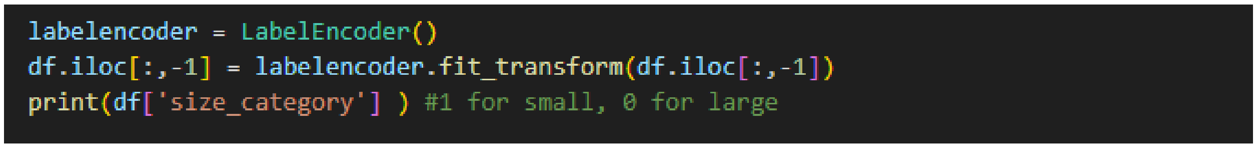
Aug month has seen highest number of small fires.

Whereas sep month has seen highest num of large fires.

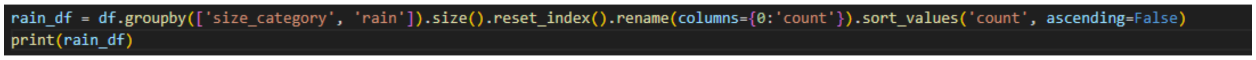
Least num of fires occured in month of nov.

Take example 2 as a reference, plot the barplot figure of the number of fires in each day(Monday to Sunday), give your observation base on the figure.

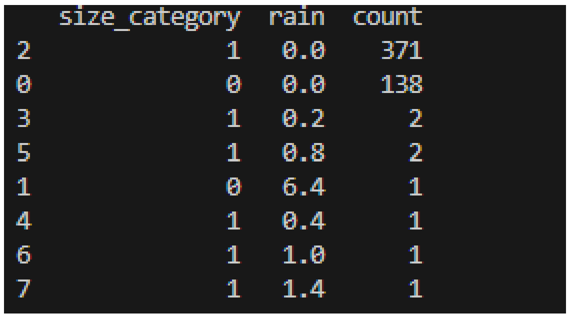
Label Encoder



```
labelencoder = LabelEncoder()
df.iloc[:,-1] = labelencoder.fit_transform(df.iloc[:,-1])
print(df['size_category'] ) #1 for small, 0 for large
```

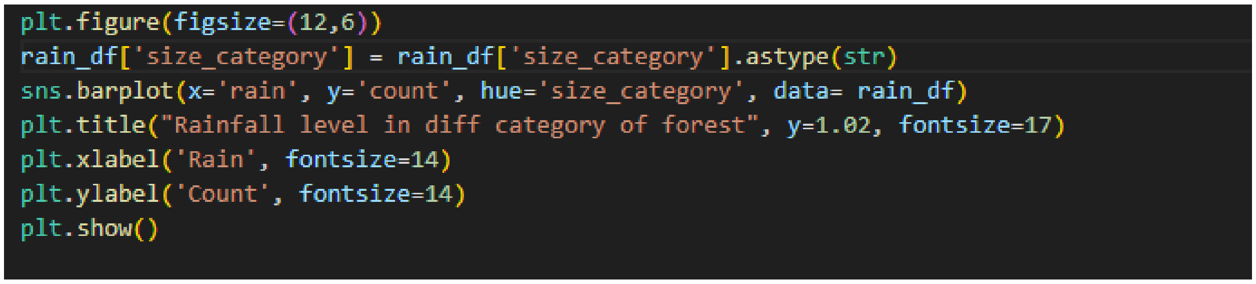


```
rain_df = df.groupby(['size_category', 'rain']).size().reset_index().rename(columns={0:'count'}).sort_values('count', ascending=False)
print(rain_df)
```

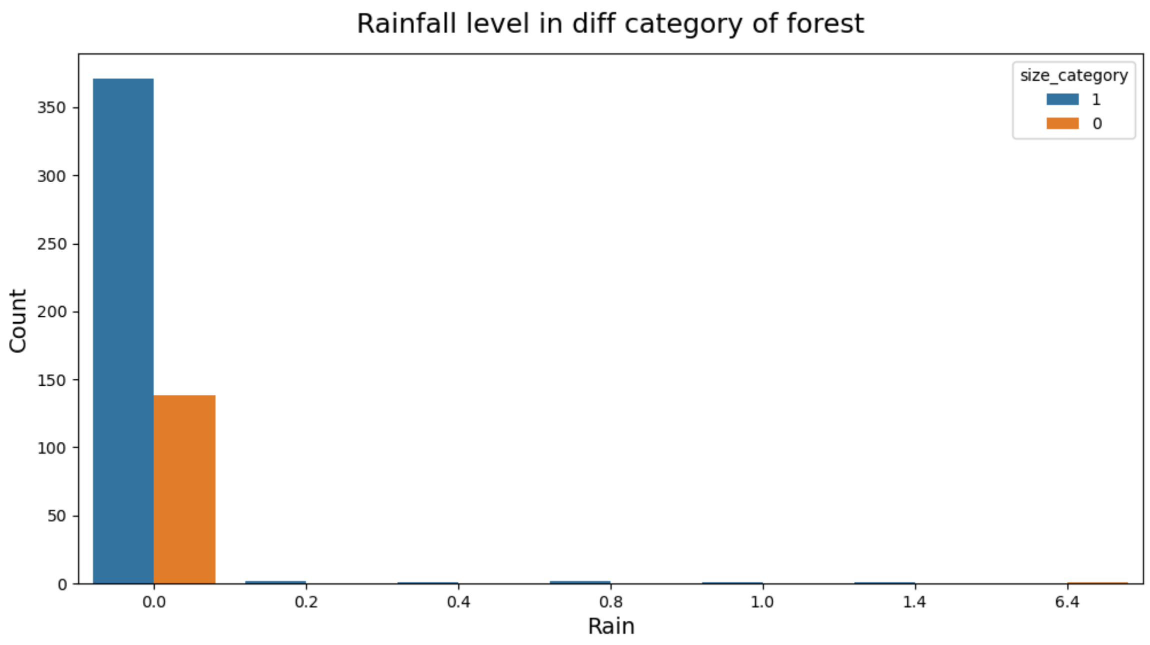


```
   size_category  rain  count
2              1   0.0    371
0              0   0.0    138
3              1   0.2      2
5              1   0.8      2
1              0   6.4      1
4              1   0.4      1
6              1   1.0      1
7              1   1.4      1
```

Example 3: rainfall level in diff category of forest



```
plt.figure(figsize=(12,6))
rain_df['size_category'] = rain_df['size_category'].astype(str)
sns.barplot(x='rain', y='count', hue='size_category', data= rain_df)
plt.title("Rainfall level in diff category of forest", y=1.02, fontsize=17)
plt.xlabel('Rain', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.show()
```

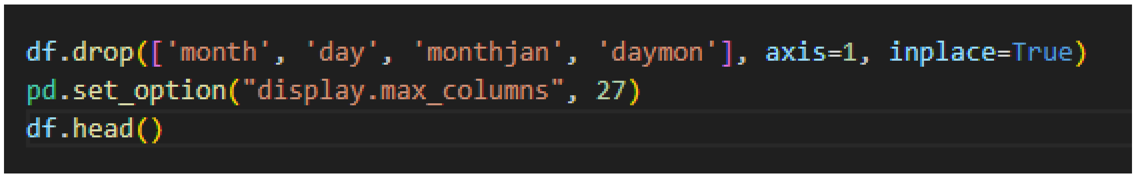


Observation:
0 represents large forest fire, 1 represents small forest fire.
Majority of the forests have almost no rainfall.
Highest rainfall a forest has was of 6.4 and it comes under a large forest fire catogory.

Take example 3 as a reference, plot the barplot figures of the RH, FFMC, DMC, DC, ISI, temp, wind, and area for size category, respectively, and give your observation base on each figure.

5. Data preprocessing

Drop the unnecessary columns



```
df.drop(['month', 'day', 'monthjan', 'daymon'], axis=1, inplace=True)
pd.set_option("display.max_columns", 27)
df.head()
```

Check the outliers

```python
from sklearn.ensemble import IsolationForest
data1=df.copy()

#training the model
clf = IsolationForest(random_state=10, contamination=.01)
clf.fit(data1)
data1['anamoly'] = clf.predict(data1.iloc[:,0:27])
outliers = data1[data1['anamoly']==-1]
print(outliers)
```

Give your observation based on the output of outliers

Remove outliers

```python
outliers.index
df.drop([281, 299, 379, 463, 464, 469], axis=0, inplace=True)
print(df.shape)
```

Splitting data into target variable and independent variables

```python
x = df.drop('size_category', axis=1)
y = df['size_category']
```

Converting independent features into normalised and standardized data

```python
norm = MinMaxScaler()
std = StandardScaler()

x_norm = pd.DataFrame(norm.fit_transform(x), columns=x.columns)   # data between -3 to +3
x_std = pd.DataFrame(std.fit_transform(x), columns=x.columns)     # data between -1 to +1
print(x_std.head())
```

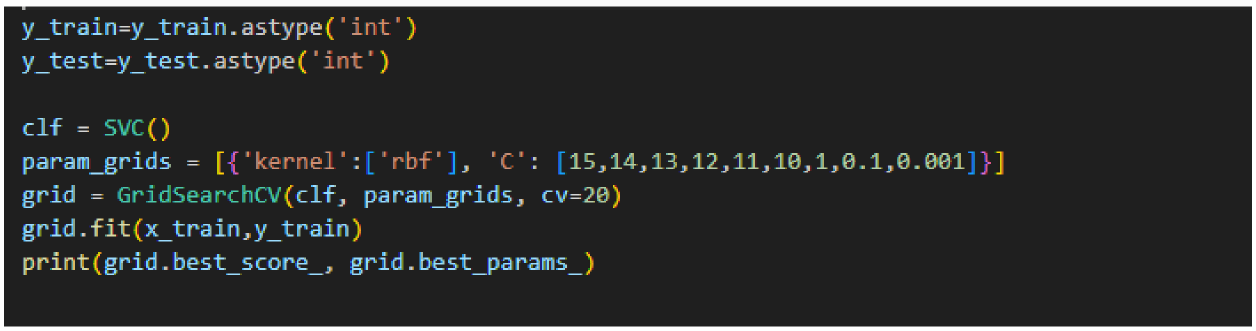
6. Model Building

Creating train and test data for model validation, with the train-test-rate = 3:1

```python
x_train, x_test, y_train, y_test = your code
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

7. Model Training | Testing | Evaluation

Kernel = rbf



```python
y_train=y_train.astype('int')
y_test=y_test.astype('int')

clf = SVC()
param_grids = [{'kernel':['rbf'], 'C': [15,14,13,12,11,10,1,0.1,0.001]}]
grid = GridSearchCV(clf, param_grids, cv=20)
grid.fit(x_train,y_train)
print(grid.best_score_, grid.best_params_)
```

Use the best C value given by the gird search and then run the below experiment, output the train and test accuracy for each experiment

rbf kernel with gamma as scale

rbf kernel with gamma as auto

Take the rbf as an example, do the grid search for the kernel of Polynomial and Linear, respectively. Use the best value given by the grid search, output the train and test train and test accuracy for each experiment

Kernel = Polynomial

Kernel = Linear

Problem Statement No 2 :
Prepare a classification model using SVM for salary data.