MACHINE LEARNING MINI-PROJECT

COURSE CODE:CSE3008

SLOT:L45+L46

Guided by:-

Dr.Srinivas Battula

Presented by:-

Valiveti Manikanta Bhuvanesh
19BCD7088

VIT-AP UNIVERSITY

ABSTRACT

USING LARGE DATASET WE HAVE TO PERFORM FIVE DIFFERENT LEARNING ALGORITHMS AND WE HAVE TO DO COMPARATIVE ANALYSIS ON THOSE FIVE ALGORITHMS

INTRODUCTION

ABOUT DATASET

I have taken health care stroke dataset with 800 instances. It contains id ,gender ,age of the patient. Whether the patient had any hyper tension and heart disease. And patients marital status ,work type ,residence type ,average glucose level , BMI ,smoking status and heart stroke status

ORIGINAL DATASET

id	gender	age	hypertensi	heart_dise	ever_marr	work_type	Residence_	avg_glucos	bmi	smoking_s	stroke
9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly s	1
51676	Female	61	0	0	Yes	Self-emplc	Rural	202.21	N/A	never smo	1
31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smo	1
60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
1665	Female	79	1	0	Yes	Self-emplc	Rural	174.12	24	never smo	1
56669	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly s	1
53882	Male	74	1	1	Yes	Private	Rural	70.09	27.4	never smo	1
10434	Female	69	0	0	No	Private	Urban	94.39	22.8	never smo	1
27419	Female	59	0	0	Yes	Private	Rural	76.15	N/A	Unknown	1
60491	Female	78	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1
12109	Female	81	1	0	Yes	Private	Rural	80.43	29.7	never smo	1
12095	Female	61	0	1	Yes	Govt_job	Rural	120.46	36.8	smokes	1
12175	Female	54	0	0	Yes	Private	Urban	104.51	27.3	smokes	1
8213	Male	78	0	1	Yes	Private	Urban	219.84	N/A	Unknown	1
5317	Female	79	0	1	Yes	Private	Urban	214.09	28.2	never smo	1
58202	Female	50	1	0	Yes	Self-emplc	Rural	167.41	30.9	never smo	1
56112	Male	64	0	1	Yes	Private	Urban	191.61	37.5	smokes	1
34120	Male	75	1	0	Yes	Private	Urban	221.29	25.8	smokes	1
27458	Female	60	0	0	No	Private	Urban	89.22	37.8	never smo	1

ABOUT PROJECT

- In this project I am going to use five algorithms, SVM linear kernel , Knn , Logistic regression, Decision tree and Random forest using single dataset.
- Using these algorithms I am going to test if patient will have stroke or not.
- And I am going to compare the f1 scores of each algorithms.

PROCEDURE

STEP 1 : DATA CLEANING

- Using data cleaning rules I have filled all the null values manually.
- Using mode operator I have filled null values in age column, work type column, bmi column and smoking status.

AFTER CLEANING DATASET

id	gender	age	hypertensi	heart_dise	ever_marr	work_type	Residence_	avg_glucos	bmi	smoking_s	stroke
9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly s	1
51676	Female	61	0	0	Yes	Self-emplc	Rural	202.21	27	never smo	1
31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smo	1
60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
1665	Female	79	1	0	Yes	Self-emplc	Rural	174.12	24	never smo	1
56669	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly s	1
53882	Male	74	1	1	Yes	Private	Rural	70.09	27.4	never smo	1
10434	Female	69	0	0	No	Private	Urban	94.39	22.8	never smo	1
27419	Female	59	0	0	Yes	Private	Rural	76.15	27	smokes	1
60491	Female	78	0	0	Yes	Private	Urban	58.57	24.2	smokes	1
12109	Female	81	1	0	Yes	Private	Rural	80.43	29.7	never smo	1
12095	Female	61	0	1	Yes	Govt_job	Rural	120.46	36.8	smokes	1
12175	Female	54	0	0	Yes	Private	Urban	104.51	27.3	smokes	1
8213	Male	78	0	1	Yes	Private	Urban	219.84	27	smokes	1
5317	Female	79	0	1	Yes	Private	Urban	214.09	28.2	never smo	1
58202	Female	50	1	0	Yes	Self-emplc	Rural	167.41	30.9	never smo	1
56112	Male	64	0	1	Yes	Private	Urban	191.61	37.5	smokes	1
34120	Male	75	1	0	Yes	Private	Urban	221.29	25.8	smokes	1
27458	Female	60	0	0	No	Private	Urban	89.22	37.8	never smo	1

STEP 2 : IMPORTING PACKAGES AND INITIALIZING DATA

Packages which I have used:

1. Pandas : Using this package we can initialize csv data and we can edit the csv file(Dataset).
2. Train_test_split: This package is from sklearn.model_slection.Using this package we can split the data into 2 parts for training and testing.
3. SVC: This package is from sklearn.svm. Using this package we can train our model by SVM classification.

4. KNeighborsClassifier: This package is from sklearn.neighbors. Using this package we can train our model by KNN classification.
5. LogisticRegression: This package is from sklearn.lineae_model. Using this package we can train our model by logistic classification.
6. DecisionTreeClassifier: This package is from sklearn.tree. Using this package we can train our model by Decision tree classification.

7. RandomForestClassifier: This package is from sklearn.ensemble. Using this package we can train our model by Random Forest classification.
8. Plot_decision_regions:This package is from mxtend.plotting.Using this package we can plot all types of classifiers graphs.
9. F1_score:This package is from sklearn.metrices.Using this package we can find the f1 score of test data using predicted values.

10. Pyplot: This package is from matplotlib. Using this package we can plot graphs.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from mlxtend.plotting import plot_decision_regions
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt
```

```
dataset = pd.read_csv('healthcare-dataset-stroke-data.csv')
print(dataset.head())
```

```
      id  gender   age  ...   bmi   smoking_status stroke
0   9046    Male  67.0  ...  36.6  formerly smoked      1
1  51676  Female  61.0  ...  27.0     never smoked      1
2  31112    Male  80.0  ...  32.5     never smoked      1
3  60182  Female  49.0  ...  34.4           smokes      1
4   1665  Female  79.0  ...  24.0     never smoked      1

[5 rows x 12 columns]
```

STEP 3 : ENCODING

- In this step we have to change the attribute values to numbers.
- For this we have to know how many unique values in each attribute . We can use unique function to find out unique values in each attribute.

```
dataset_name.attribute_name.unique()
```

- With the above syntax we can find unique values.

```python
print(dataset.smoking_status.unique())
print(dataset.gender.unique())
print(dataset.hypertension.unique())
print(dataset.heart_disease.unique())
print(dataset.ever_married.unique())
print(dataset.Residence_type.unique())
print(dataset.work_type.unique())
```

```
['formerly smoked' 'never smoked' 'smokes']
['Male' 'Female']
[0 1]
[1 0]
['Yes' 'No']
['Urban' 'Rural']
['Private' 'Self-employed' 'Govt_job' 'Never_worked']
```

- Using unique values we have to change that values into number format instead of names.
- First we need to create a dictionary according to unique values we have got.
- Next we have to map that dictionary to the attribute values and assign to a temporary variable.
- Then we have to change the attribute values by temporary variable.

```python
smoking_status_to_num = {'never smoked': 0,'smokes': 1,'formerly smoked': 2}
temp = dataset['smoking_status'].map(smoking_status_to_num)
print(temp)
dataset['smoking_status']=temp
```

```
0      2
1      0
2      0
3      1
4      0
      ..
794    1
795    2
796    0
797    0
798    1
Name: smoking_status, Length: 799, dtype: int64
```

```python
Residence_type_to_num={'Urban':0,'Rural':1}
temp1=dataset['Residence_type'].map(Residence_type_to_num)
print(temp1)
dataset['Residence_type']=temp1
```

```
0      0
1      1
2      1
3      0
4      1
      ..
794    1
795    1
796    0
797    0
798    0
Name: Residence_type, Length: 799, dtype: int64
```

```python
work_type_to_num={'Private':0,'Self-employed':1,'Govt_job':2,'Never_worked':3}
temp2=dataset['work_type'].map(work_type_to_num)
print(temp2)
dataset['work_type']=temp2
```

```
0      0
1      1
2      0
3      0
4      1
      ..
794    0
795    2
796    0
797    0
798    0
Name: work_type, Length: 799, dtype: int64
```

```python
gender_to_num={'Male':0,'Female':1}
temp3=dataset['gender'].map(gender_to_num)
print(temp3)
dataset['gender']=temp3
```

```
0      0
1      1
2      0
3      1
4      1
      ..
794    1
795    1
796    1
797    1
798    1
Name: gender, Length: 799, dtype: int64
```

```python
ever_married_to_num={'Yes':0,'No':1}
temp4=dataset['ever_married'].map(ever_married_to_num)
print(temp4)
dataset['ever_married']=temp4
```

```
0      0
1      0
2      0
3      0
4      0
      ..
794    0
795    0
796    1
797    1
798    1
Name: ever_married, Length: 799, dtype: int64
```

```python
print(dataset)
```

```
     gender   age  hypertension  ...   bmi  smoking_status  stroke
0         0  67.0             0  ...  36.6               2       1
1         1  61.0             0  ...  27.0               0       1
2         0  80.0             0  ...  32.5               0       1
3         1  49.0             0  ...  34.4               1       1
4         1  79.0             1  ...  24.0               0       1
..      ...   ...           ...  ...   ...             ...     ...
794       1  42.0             0  ...  24.0               1       0
795       1  31.0             0  ...  20.5               2       0
796       1  29.0             0  ...  20.4               0       0
797       1  23.0             0  ...  27.6               0       0
798       1  25.0             0  ...  27.0               1       0
```

STEP 4 : SPLITTING DATA FOR TEST AND TRAIN

- First we need to split data into two variables .One is for X values and Y is for target values(Stroke).
- Using drop function we assigned X values.
- Using train split function we can split the X and Y values to two parts by assigning test size we need.
- By this we can get values for training and testing.

```python
X=dataset.drop('stroke',axis=1)
print(X)

Y=dataset['stroke']
print(Y)
```

```
     gender   age  hypertension  ...  avg_glucose_level   bmi  smoking_status
0         0  67.0             0  ...             228.69  36.6               2
1         1  61.0             0  ...             202.21  27.0               0
2         0  80.0             0  ...             105.92  32.5               0
3         1  49.0             0  ...             171.23  34.4               1
4         1  79.0             1  ...             174.12  24.0               0
..      ...   ...           ...  ...                ...   ...             ...
794       1  42.0             0  ...              88.31  24.0               1
795       1  31.0             0  ...             103.55  20.5               2
796       1  29.0             0  ...              60.26  20.4               0
797       1  23.0             0  ...              79.39  27.6               0
798       1  25.0             0  ...             122.01  27.0               1

[799 rows x 10 columns]
0      1
1      1
2      1
3      1
4      1
      ..
794    0
795    0
796    0
797    0
798    0
Name: stroke, Length: 799, dtype: int64
```

```python
X1=dataset.loc[:,['age','bmi']]
X1=X1.to_numpy()
Y1=Y.to_numpy()
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2)
```

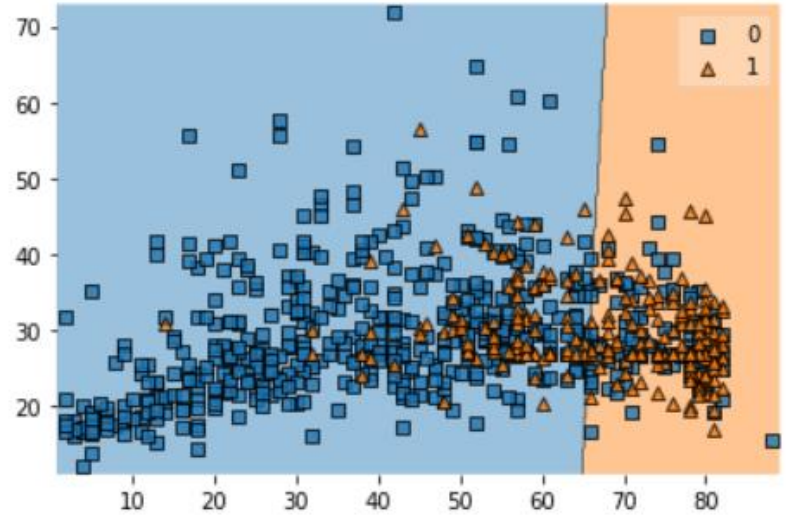
STEP 5 : TRAINING OF DATA

SUPPORT VECTOR MACHINE(SVM)

- For SVM training we have to initialize the classifier.
- Next we have to fit the train data X_train,Y_train into the classifier.
- Then we can plot the classifier using plot_decision_region function.
- For f1 score we have to predict X_test and then we compare with the Y_test then we can get the f1 score of train dataset.

```
model = SVC(kernel='linear',C=1)
model.fit(X1,Y1)
plot_decision_regions(X=X1,y=Y1,clf=model)
```

```
/usr/local/lib/python3.7/dist-packages/mlxtend/plotting/decisi
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.min(), y_max=
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a8facaad0>
```




```
model=SVC(kernel='linear')
model.fit(X_train,Y_train)
predict=model.predict(X_test)
SVM_f1=f1_score(Y_test,predict)
SVM_f1
```

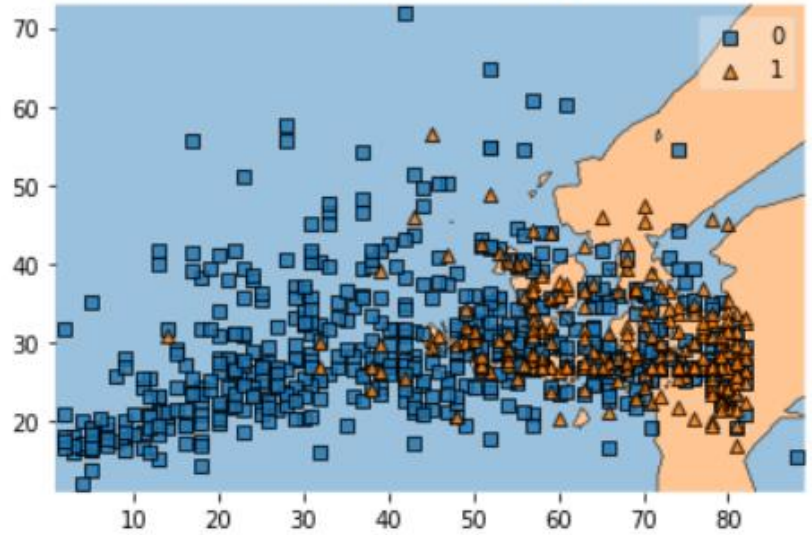
```
0.6382978723404256
```

K-NEAREST NEIGHBORS (KNN)

- For KNN training we have to initialize the classifier.
- Next we have to fit the train data X_train,Y_train into the classifier.
- Then we can plot the classifier using plot_decision_region function.
- For f1 score we have to predict X_test and then we compare with the Y_test then we can get the f1 score of train dataset.

```
knn=KNeighborsClassifier()
knn.fit(X1,Y1)
plot_decision_regions(X=X1,y=Y1,clf=knn)
```

```
/usr/local/lib/python3.7/dist-packages/mlxtend/plotti
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.min(
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a8fa36
```




```
knn=KNeighborsClassifier()
knn.fit(X_train,Y_train)
predict=knn.predict(X_test)
knn_f1=f1_score(Y_test,predict)
knn_f1
```

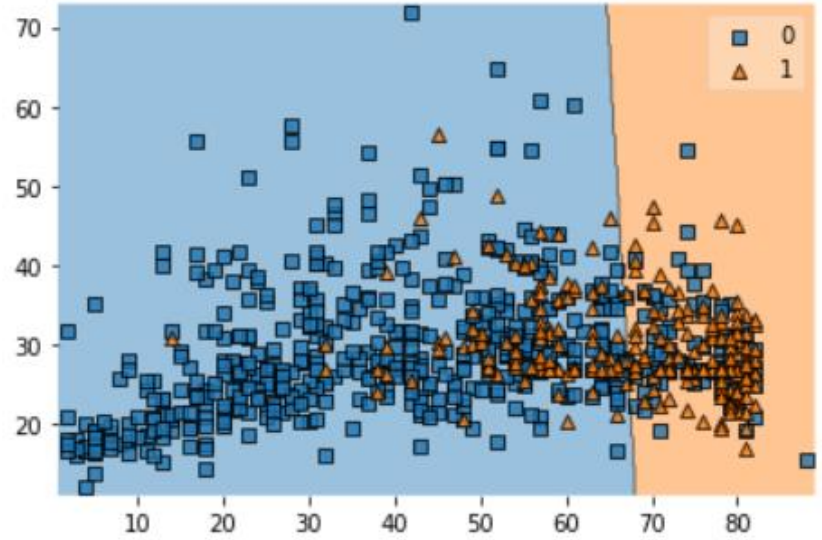
```
0.625
```

LOGISTIC REGRESSION

- For logistic regression training we have to initialize the classifier.
- Next we have to fit the train data X_train,Y_train into the classifier.
- Then we can plot the classifier using plot_decision_region function.
- For f1 score we have to predict X_test and then we compare with the Y_test then we can get the f1 score of train dataset.

```
lr=LogisticRegression()
lr.fit(X1,Y1)
plot_decision_regions(X=X1,y=Y1,clf=lr)
```

```
/usr/local/lib/python3.7/dist-packages/mlxtend/plott
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.mir
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a8f9c
```




```
lr=LogisticRegression()
lr.fit(X_train,Y_train)
predict=lr.predict(X_test)
lr_f1=f1_score(Y_test,predict)
lr_f1
```

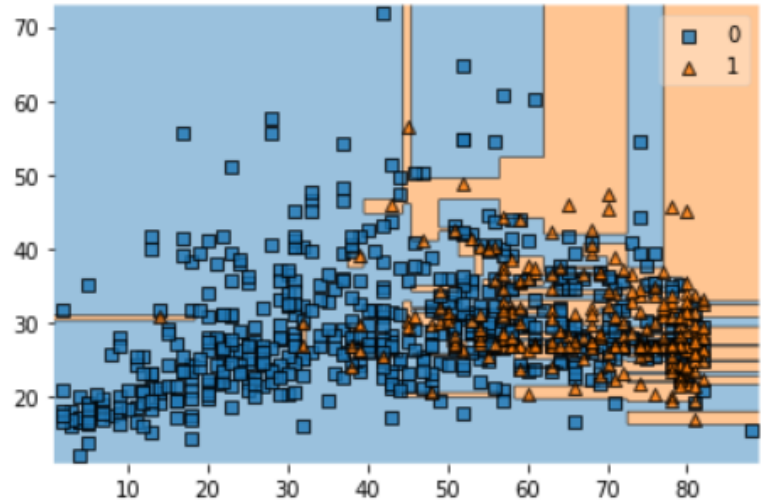
```
/usr/local/lib/python3.7/dist-p
STOP: TOTAL NO. of ITERATIONS R

Increase the number of iteratio
    https://scikit-learn.org/st
Please also refer to the docume
    https://scikit-learn.org/st
  extra_warning_msg=_LOGISTIC_S
0.6813186813186813
```

DECISION TREE

- For decision tree training we have to initialize the classifier.
- Next we have to fit the train data X_train,Y_train into the classifier.
- Then we can plot the classifier using plot_decision_region function.
- For f1 score we have to predict X_test and then we compare with the Y_test then we can get the f1 score of train dataset.

```
dt=DecisionTreeClassifier()
dt.fit(X1,Y1)
plot_decision_regions(X=X1,y=Y1,clf=dt)
```

```
/usr/local/lib/python3.7/dist-packages/mlxtend/plot
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.mi
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a8fa
```




```
dt=DecisionTreeClassifier()
dt.fit(X_train,Y_train)
predict=dt.predict(X_test)
dt_f1=f1_score(Y_test,predict)
dt_f1
```

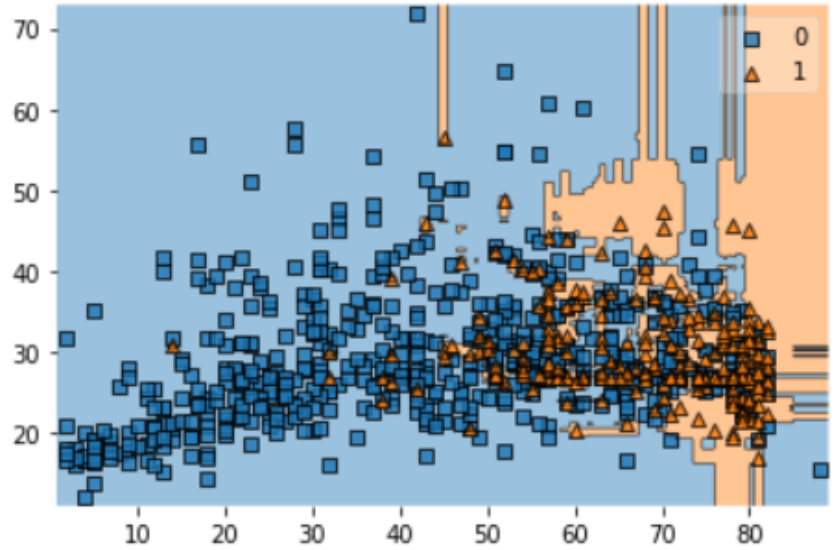
```
0.5473684210526316
```

RANDOM FOREST

- For random forest training we have to initialize the classifier.
- Next we have to fit the train data X_train,Y_train into the classifier.
- Then we can plot the classifier using plot_decision_region function.
- For f1 score we have to predict X_test and then we compare with the Y_test then we can get the f1 score of train dataset.

```
rf=RandomForestClassifier()
rf.fit(X1,Y1)
plot_decision_regions(X=X1,y=Y1,clf=rf)
```

```
/usr/local/lib/python3.7/dist-packages/mlxtend/plc
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.m
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a8f
```



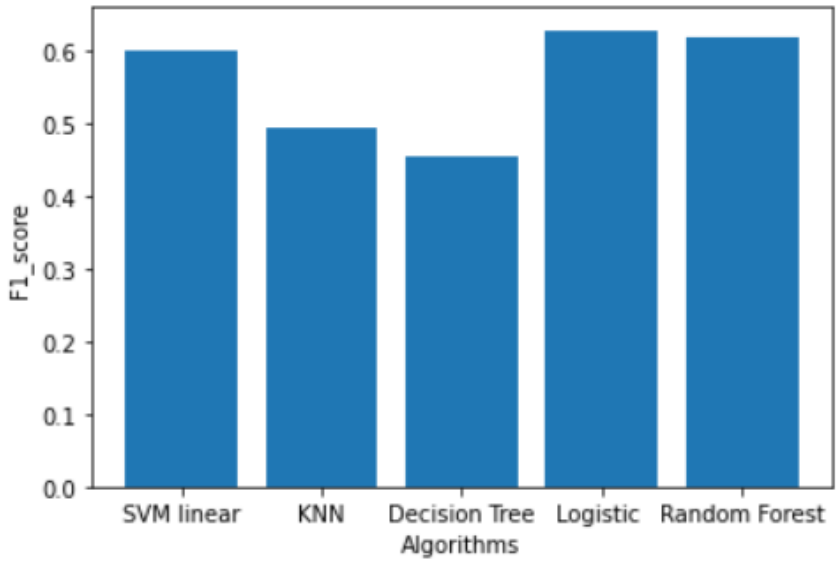

```
rf=RandomForestClassifier()
rf.fit(X_train,Y_train)
predict=rf.predict(X_test)
rf_f1=f1_score(Y_test,predict)
rf_f1
```

```
0.6382978723404256
```

OBSERVATION

- I have taken all the f1 score values and assigned to variables.
- Using the pyplot function I plotted all the f1 score values in a bar graph.
- I have labelled Algorithms on x-axis and f1 scores on y-axis.

```
plt.bar(['SVM linear','KNN','Decision Tree','Logistic ','Random Forest'],[SVM_f1,knn_f1,dt_f1,lr_f1,rf_f1])
plt.xlabel("Algorithms")
plt.ylabel("F1_score")
plt.show()
```



CONCLUSION

I conclude that for every new runtime f1 scores of each algorithms are varying approximately 10-20%.With this f1 scores we can say how much percentage of test data is correctly predicted by train data.

the End

dreamstime