In [2]:
```python
import pandas as pd
```

In [3]:
```python
data=pd.read_csv("C:\\Users\\vaibhav vishal\\Downloads\\diabetes.csv")
```

In [4]:
```python
data.head()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct
0	6	148	72	35	0	33.6	0.
1	1	85	66	29	0	26.6	0.
2	8	183	64	0	0	23.3	0.
3	1	89	66	23	94	28.1	0.
4	0	137	40	35	168	43.1	2.

In [5]:
```python
data.tail()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

In [6]:
```python
data.shape
```

Out[6]:
```
(768, 9)
```

In [7]:
```python
print("number of rows",data.shape[0])
print("number of columns",data.shape[1])
```

```
number of rows 768
number of columns 9
```

#checkin null values in dataset

In [8]:
```python
data.isnull().sum()
```

Out[8]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [9]:
```
#checking overall statistics
data.describe()
```

Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diab
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [10]:
```
import numpy as np
```

In [11]:
```
data_copy=data.copy(deep=True)
```

In [12]:
```
data.columns
```

Out[12]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insuli
n',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [13]:
```
data_copy[[ 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI']]=data_copy[[ 'Glucose', 'BloodPressure', 'SkinThickness', 'In
       'BMI']].replace(0,np.nan)
```

In [14]:
```
data_copy.isnull().sum()
```

Out[14]:
```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

In [15]:
```
data['Glucose'] = data['Glucose'].replace(0, data['Glucose'].mean())
data['SkinThickness'] = data['SkinThickness'].replace(0, data['SkinThicknes
data['Insulin'] = data['Insulin'].replace(0, data['Insulin'].mean())
data['BMI'] = data['BMI'].replace(0, data['BMI'].mean())
data['Pregnancies'] = data['Pregnancies'].replace(0, data['Pregnancies'].me
```

In [16]:
```
X=data.drop('Outcome',axis=1)
Y=data['Outcome']
```

In [17]:
```
#splitting the datasset into training set and 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=0.20,random_st
In [40]: print(X.shape,X_train.shape,X_test.shape)
         print(X_train,X_test,Y_train,Y_test)
```

```
(768, 8) (614, 8) (154, 8)
     Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin  \
60      2.000000     84.0              0      20.536458   79.799479   
618     9.000000    112.0             82      24.000000   79.799479   
346     1.000000    139.0             46      19.000000   83.000000   
294     3.845052    161.0             50      20.536458   79.799479   
231     6.000000    134.0             80      37.000000  370.000000   
..           ...      ...            ...            ...         ...   
71      5.000000    139.0             64      35.000000  140.000000   
106     1.000000     96.0            122      20.536458   79.799479   
270    10.000000    101.0             86      37.000000   79.799479   
435     3.845052    141.0              0      20.536458   79.799479   
102     3.845052    125.0             96      20.536458   79.799479   

           BMI  DiabetesPedigreeFunction  Age  
60   31.992578                     0.304   21  
618  28.200000                     1.282   50  
346  28.700000                     0.654   22  
294  21.900000                     0.254   65  
231  46.200000                     0.238   46  
..         ...                       ...  ...  
71   28.600000                     0.411   26  
106  22.400000                     0.207   27  
270  45.600000                     1.136   38  
435  42.400000                     0.205   29  
102  22.500000                     0.262   21  

[614 rows x 8 columns]      Pregnancies  Glucose  BloodPressure  SkinThick
ness     Insulin   BMI  \
668          6.0     98.0             58      33.000000  190.000000  34.0   
324          2.0    112.0             75      32.000000   79.799479  35.7   
624          2.0    108.0             64      20.536458   79.799479  30.8   
690          8.0    107.0             80      20.536458   79.799479  24.6   
473          7.0    136.0             90      20.536458   79.799479  29.9   
..           ...      ...            ...            ...         ...   ...   
355          9.0    165.0             88      20.536458   79.799479  30.4   
534          1.0     77.0             56      30.000000   56.000000  33.3   
344          8.0     95.0             72      20.536458   79.799479  36.8   
296          2.0    146.0             70      38.000000  360.000000  28.0   
462          8.0     74.0             70      40.000000   49.000000  35.3   

     DiabetesPedigreeFunction  Age  
668                     0.430   43  
324                     0.148   21  
624                     0.158   21  
690                     0.856   34  
473                     0.210   50  
..                        ...  ...  
355                     0.302   49  
534                     1.251   24  
344                     0.485   57  
296                     0.337   29  
462                     0.705   39  

[154 rows x 8 columns] 60     0
618    1
346    0
294    0
231    1
      ..
71     0
```

```
106    0
270    1
435    1
102    0
Name: Outcome, Length: 614, dtype: int64 668    0
324    0
624    0
690    0
473    0
      ..
355    1
534    0
344    0
296    1
462    0
Name: Outcome, Length: 154, dtype: int64
```

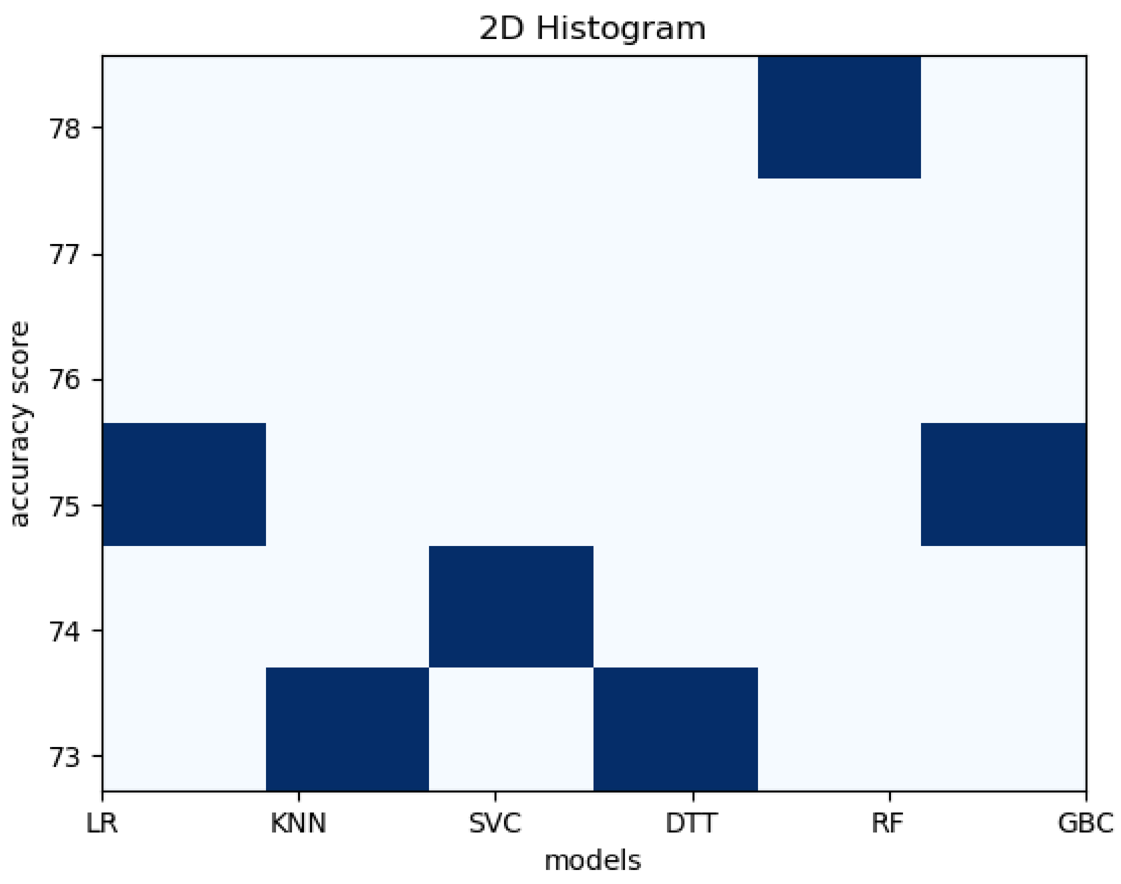
In [19]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.pipeline import Pipeline
```

In [20]:
```
pipeline_lr=Pipeline([('scalar1',StandardScaler()),
                      ('lr_classifier',LogisticRegression())])
pipeline_knn=Pipeline([('scalar2',StandardScaler()),
                       ('knn_classifier',KNeighborsClassifier())])
pipeline_svc=Pipeline([('scaler3',StandardScaler()),
                       ('svc_classifier',SVC())])
pipeline_dt=Pipeline([('dt_classifier',DecisionTreeClassifier())])
pipeline_rf=Pipeline([('rf_classifier',RandomForestClassifier(max_depth=3))
pipeline_gbc=Pipeline([('gbc_classifier',GradientBoostingClassifier())])
#pipeline_vt=Pipeline([('voting_classifier',VotingClassifier(estimators=[(p
```

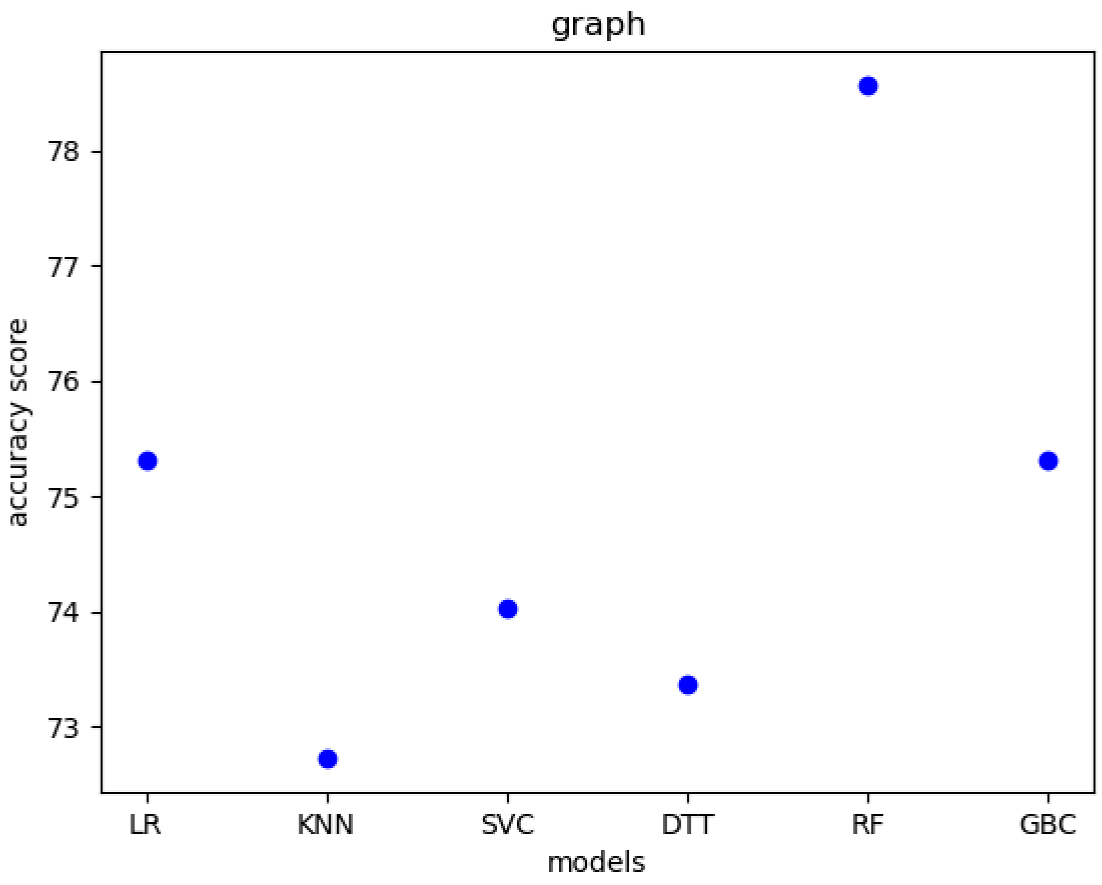
In [21]:
```
pipelines=[pipeline_lr,
           pipeline_knn,
           pipeline_svc,
           pipeline_dt,
           pipeline_rf,
           pipeline_gbc]
```

In [22]:
```
pipelines
```
Out[22]:
```
[Pipeline(steps=[('scalar1', StandardScaler()),
                 ('lr_classifier', LogisticRegression())]),
 Pipeline(steps=[('scalar2', StandardScaler()),
                 ('knn_classifier', KNeighborsClassifier())]),
 Pipeline(steps=[('scaler3', StandardScaler()), ('svc_classifier', SVC
())]),
 Pipeline(steps=[('dt_classifier', DecisionTreeClassifier())]),
 Pipeline(steps=[('rf_classifier', RandomForestClassifier(max_depth=3))]),
 Pipeline(steps=[('gbc_classifier', GradientBoostingClassifier())])]
```
In [23]:
```
for pipe in pipelines:
    pipe.fit(X_train,Y_train)
```
In [24]:
```
pipe_dict={0:'LR',
           1:'KNN',
           2:'SVC',
           3:'DT',
           4:'RF',
           5:'GBC'}
```
In [25]:
```
pipe_dict
```
Out[25]:
```
{0: 'LR', 1: 'KNN', 2: 'SVC', 3: 'DT', 4: 'RF', 5: 'GBC'}
```
In [26]:
```
for i,model in enumerate(pipelines):
    print("{}Test Accuracy:{}".format(pipe_dict[i],model.score(X_test,Y_tes
```
```
LRTest Accuracy:75.32467532467533
KNNTest Accuracy:72.72727272727273
SVCTest Accuracy:74.02597402597402
DTTest Accuracy:73.37662337662337
RFTest Accuracy:77.27272727272727
GBCTest Accuracy:75.32467532467533
```
In [27]:
```
import matplotlib.pyplot as plt
```

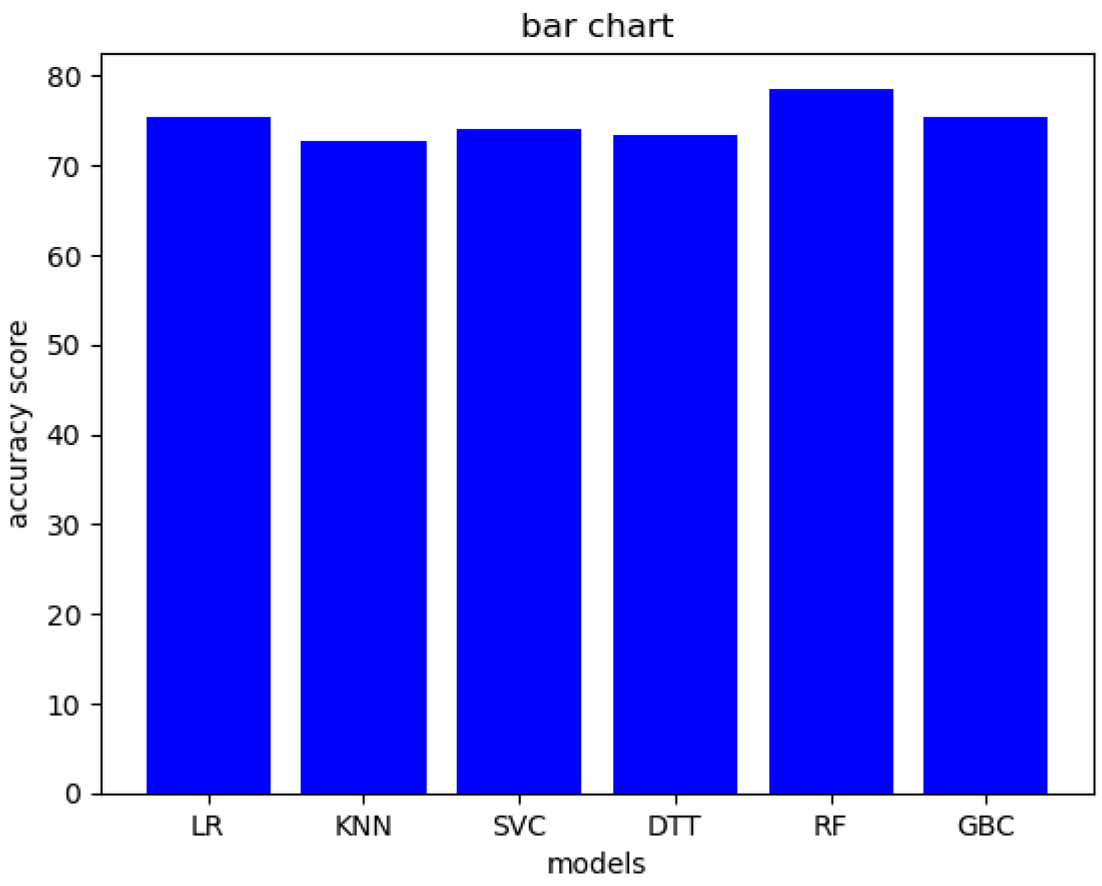
In [28]:
```
x=np.array([0,1,2,3,4,5])
y=np.array([75.3246,72.7272,74.0259,73.3766,78.5714,75.3246])
plt.hist2d(x,y, bins=(6,6), cmap=plt.cm.Blues)
plt.xticks([0,1,2,3,4,5],['LR','KNN','SVC','DTT','RF','GBC'])
plt.xlabel('models')
plt.ylabel('accuracy score')
plt.title('2D Histogram')
plt.show()
```



In [29]:
```
x=np.array([0,1,2,3,4,5])
y=np.array([75.3246,72.7272,74.0259,73.3766,78.5714,75.3246])
plt.plot(x,y,'o',color='blue')
plt.xticks([0,1,2,3,4,5],['LR','KNN','SVC','DTT','RF','GBC'])
plt.xlabel('models')
plt.ylabel('accuracy score')
plt.title('graph')
plt.show()
```



In [30]:
```
x=['LR','KNN','SVC','DTT','RF','GBC']
y=[75.3246,72.7272,74.0259,73.3766,78.5714,75.3246]
plt.bar(x,y,color='blue')
plt.xlabel('models')
plt.ylabel('accuracy score')
plt.title('bar chart')
plt.show()
```



In [31]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [32]:
```
X=data.drop('Outcome',axis=1)
Y=data['Outcome']
```

In [33]:
```
rf=RandomForestClassifier(max_depth=3)
```

In [34]:
```
rf.fit(X,Y)
```

Out[34]:
```
RandomForestClassifier(max_depth=3)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [35]:
```
feature_importances=rf.feature_importances_
for i,feature in enumerate(X.columns):
    print(f"{feature}:{feature_importances[i]*100}")
```

```
Pregnancies:10.041377590538454
Glucose:35.730179879623385
BloodPressure:1.6611141066173896
SkinThickness:4.99578294452019
Insulin:7.302576836746192
BMI:16.297431208069245
DiabetesPedigreeFunction:8.00721601873113
Age:15.964321415154012
```

In [37]:
```
#prediction on new data
new_data=pd.DataFrame({
    'Pregnancies':6,
    'Glucose':148.0,
    'BloodPressure':72.0,
    'SkinThickness':35.0,
    'Insulin':0.0,
    'BMI':33.6,
    'DiabetesPedigreeFunction':0.627,
    'Age':50
},index=[0])
```

In [38]:
```
p=rf.predict(new_data)
```

In [39]:
```
if p[0]==0:
    print('non-diabetic')
else:
    print('diabetic')
```

```
diabetic
```

In [48]:
```
#save the model
import joblib
```

In [49]:
```
joblib.dump(rf,'diabetes_model')
```

Out[49]:
```
['diabetes_model']
```

In [50]:
```
model=joblib.load('diabetes_model')
```

In [51]:
```
model.predict(new_data)
```

Out[51]:
```
array([1], dtype=int64)
```

In [52]:
```
#creating gui
from tkinter import *
import joblib
```

In [ ]:
```python
from tkinter import *
import joblib
import numpy as np
from sklearn import *
def show_entry_fields():
    p1=float(e1.get())
    p2=float(e2.get())
    p3=float(e3.get())
    p4=float(e4.get())
    p5=float(e5.get())
    p6=float(e6.get())
    p7=float(e7.get())
    p8=float(e8.get())

    model=joblib.load('diabetes_model')
    result=model.predict([[p1,p2,p3,p4,p5,p6,p7,p8]])

    if result==0:
        Label(master, text="Non-Diabetic").grid(row=31)
    else:
        Label(master, text="Diabetic").grid(row=31)

master = Tk()
master.title("Diabetes Prediction Using Machine Learning")


label=Label(master,text="Diabetes Prediction using Machine Learning"
                       ,bg="yellow",fg="red"). \
                           grid(row=0,columnspan=2)


Label(master, text="Pregnancies").grid(row=1)
Label(master, text="Glucose").grid(row=2)
Label(master, text="Enter value of BloodPressure").grid(row=3)
Label(master, text="Enter value of SkinThickness").grid(row=4)
Label(master, text="Enter value of Insulin").grid(row=5)
Label(master, text="Enter value of BMI").grid(row=6)
Label(master, text="Enter value of DiabetesPedigreeFunction").grid(row=7)
Label(master, text="Enter value of Age").grid(row=8)


e1=Entry(master)
e2=Entry(master)
e3=Entry(master)
e4=Entry(master)
e5=Entry(master)
e6=Entry(master)
e7=Entry(master)
e8=Entry(master)

e1.grid(row=1,column=1)
e2.grid(row=2,column=1)
e3.grid(row=3,column=1)
e4.grid(row=4,column=1)
e5.grid(row=5,column=1)
e6.grid(row=6,column=1)
e7.grid(row=7,column=1)
e8.grid(row=8,column=1)

Button(master, text='predict',command=show_entry_fields).grid()
```

```
mainloop()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: