DIABETES PREDICTION

- Algorithm: Random Forest Classifier, K-Nearest Neighbors (KNN), adaboosting classifier,linear Regression, Decision Tree Classifier,
- Description: Predict whether a person has diabetes or not using features like glucose levels and BMI.

In [187…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import warnings
warnings.filterwarnings('ignore')
```

In [188…
```
df=pd.read_csv('diabetes_prediction_dataset.csv')
df
```

Out[188]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0
...	...	...	...	...	...	...	...	...	...
99995	Female	80.0	0	0	No Info	27.32	6.2	90	0
99996	Female	2.0	0	0	No Info	17.37	6.5	100	0
99997	Male	66.0	0	0	former	27.83	5.7	155	0
99998	Female	24.0	0	0	never	35.42	4.0	100	0
99999	Female	57.0	0	0	current	22.43	6.6	90	0

100000 rows × 9 columns

In [189…
```
df.head()
```

Out[189]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0

In [190…
```
df.tail()
```

Out[190]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
99995	Female	80.0	0	0	No Info	27.32	6.2	90	0
99996	Female	2.0	0	0	No Info	17.37	6.5	100	0
99997	Male	66.0	0	0	former	27.83	5.7	155	0
99998	Female	24.0	0	0	never	35.42	4.0	100	0
99999	Female	57.0	0	0	current	22.43	6.6	90	0

In [191…
```
df.isnull().sum()
```

Out[191]:
```
gender                 0
age                    0
hypertension           0
heart_disease          0
smoking_history        0
bmi                    0
HbA1c_level            0
blood_glucose_level    0
diabetes               0
dtype: int64
```

In [192…
```python
print("Number of rows =",df.shape[0])
print("Number of columns =",df.shape[1])
```

```
Number of rows = 100000
Number of columns = 9
```

In [193…
```python
df.size
```

Out[193]:
```
900000
```

In [194…
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   gender               100000 non-null  object 
 1   age                  100000 non-null  float64
 2   hypertension         100000 non-null  int64  
 3   heart_disease        100000 non-null  int64  
 4   smoking_history      100000 non-null  object 
 5   bmi                  100000 non-null  float64
 6   HbA1c_level          100000 non-null  float64
 7   blood_glucose_level  100000 non-null  int64  
 8   diabetes             100000 non-null  int64  
dtypes: float64(3), int64(4), object(2)
memory usage: 6.9+ MB
```

In [195…
```python
df.describe()
```

Out[195]:

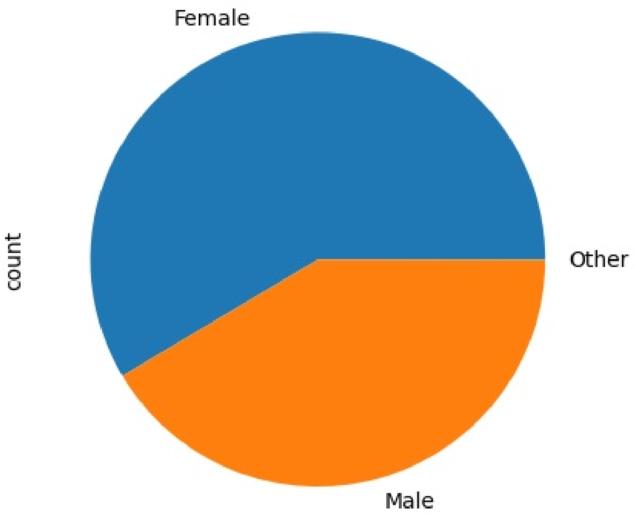
	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes
count	100000.000000	100000.00000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000
mean	41.885856	0.07485	0.039420	27.320767	5.527507	138.058060	0.085000
std	22.516840	0.26315	0.194593	6.636783	1.070672	40.708136	0.278883
min	0.080000	0.00000	0.000000	10.010000	3.500000	80.000000	0.000000
25%	24.000000	0.00000	0.000000	23.630000	4.800000	100.000000	0.000000
50%	43.000000	0.00000	0.000000	27.320000	5.800000	140.000000	0.000000
75%	60.000000	0.00000	0.000000	29.580000	6.200000	159.000000	0.000000
max	80.000000	1.00000	1.000000	95.690000	9.000000	300.000000	1.000000

In [196…
```python
df['gender'].value_counts()
```

Out[196]:
```
gender
Female    58552
Male      41430
Other        18
Name: count, dtype: int64
```

In [197…
```python
df['gender'].value_counts().plot(kind='pie')
```

Out[197]:
```
<Axes: ylabel='count'>
```



In [198…

```
df['smoking_history'].value_counts()
```

Out[198]:

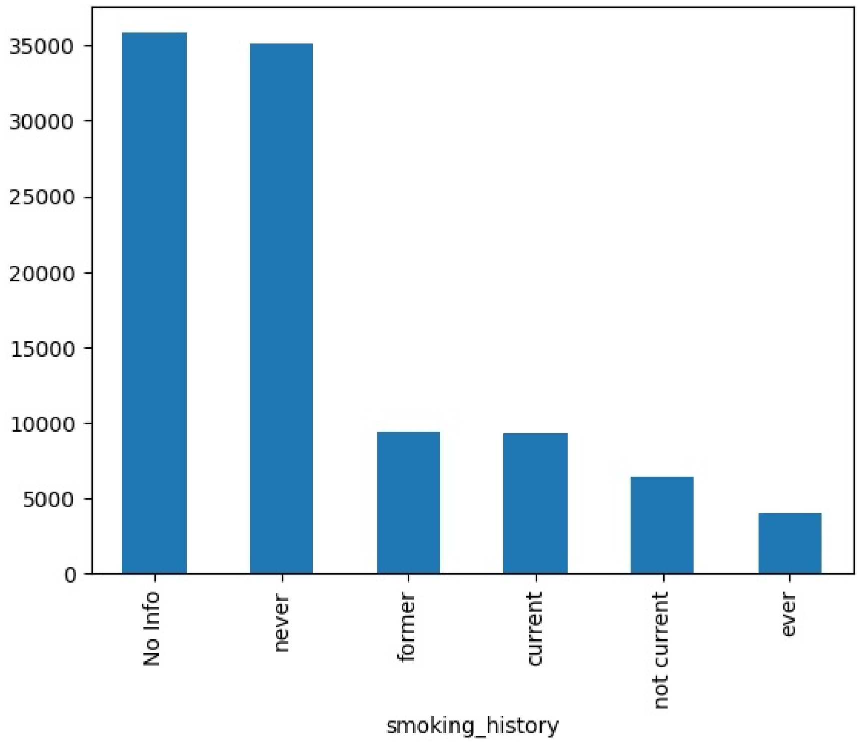
```
smoking_history
No Info        35816
never          35095
former          9352
current         9286
not current     6447
ever            4004
Name: count, dtype: int64
```

In [199…

```
df['smoking_history'].value_counts().plot(kind='bar')
```

Out[199]:

```
<Axes: xlabel='smoking_history'>
```



In [200…
```
df.columns
```
Out[200]:
```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'smoking_history',
       'bmi', 'HbA1c_level', 'blood_glucose_level', 'diabetes'],
      dtype='object')
```

In [201…
```
#Convert categorical feature into numerical feature by using label encoding technique and other technique using
encode = LabelEncoder()
df['smoking_history']=encode.fit_transform(df['smoking_history'])
df['gender']=np.where(df['gender'].str.contains('Female'),0,1)
```

In [202…
```
df.head()
```
Out[202]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	0	80.0	0	1	4	25.19	6.6	140	0
1	0	54.0	0	0	0	27.32	6.6	80	0
2	1	28.0	0	0	4	27.32	5.7	158	0
3	0	36.0	0	0	1	23.45	5.0	155	0
4	1	76.0	1	1	1	20.14	4.8	155	0

In [203…
```
#Take relevant features from the dataset to train our model.
x = df[['smoking_history','gender','bmi','blood_glucose_level']]
```

```python
y = df['diabetes']
```

In [204…
```python
#Doing standard scaling
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
```

In [205…
```python
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.2, random_state=42)
```

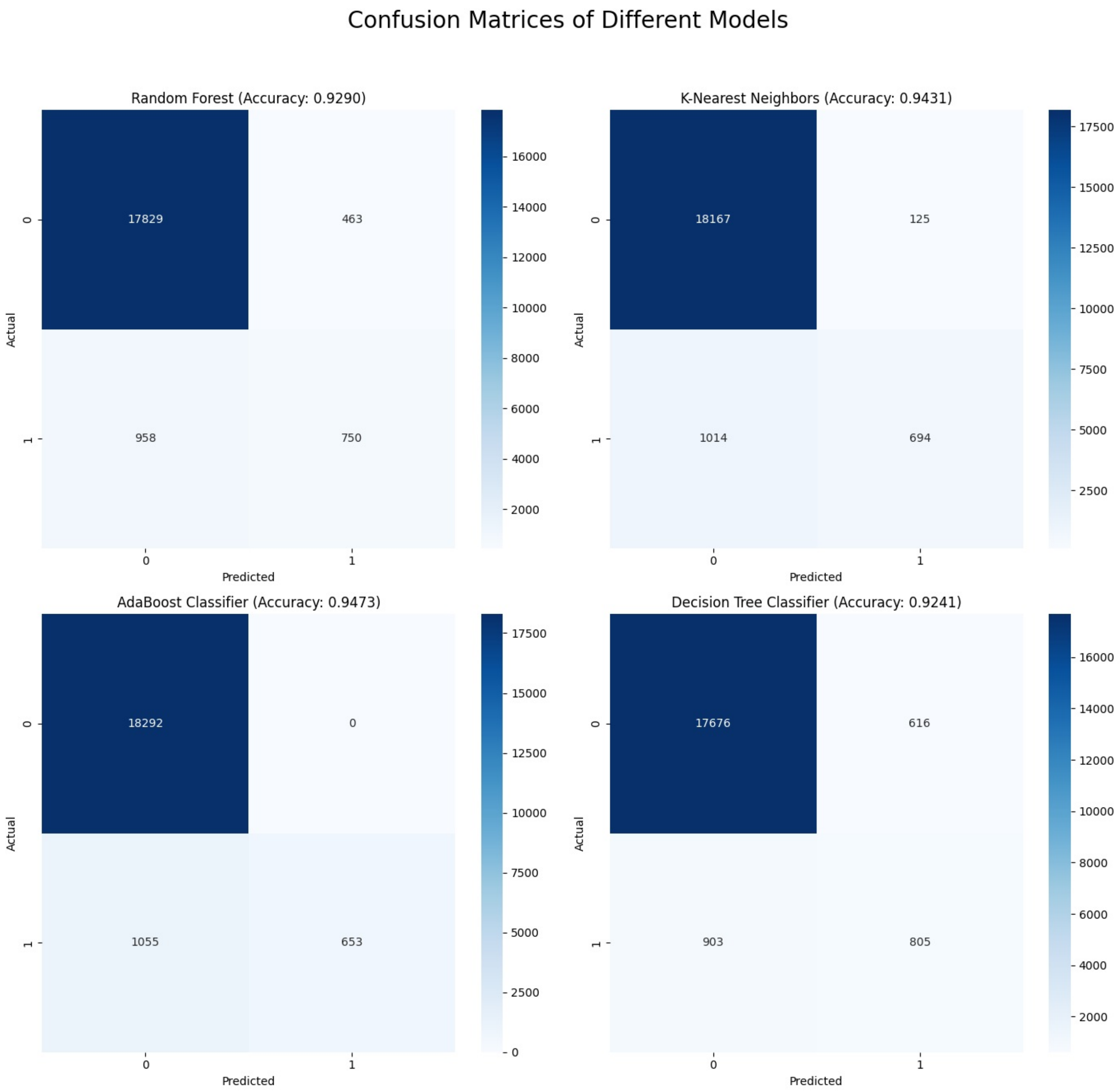
In [206…
```python
# Define models
models = [
    ("Random Forest", RandomForestClassifier(random_state=42)),
    ("K-Nearest Neighbors", KNeighborsClassifier()),
    ("AdaBoost Classifier", AdaBoostClassifier(random_state=42)),
    ("Decision Tree Classifier", DecisionTreeClassifier(random_state=42)),
    ("Logistic Regression", LogisticRegression(random_state=42))
]
```

In [207…
```python
# Train and evaluate models, storing results
results = {}
for name, model in models:
    model.fit(x_train, y_train)
    predictions = model.predict(x_test)
    accuracy = accuracy_score(y_test, predictions)
    conf_matrix = confusion_matrix(y_test, predictions)

    results[name] = {
        "Accuracy": accuracy,
        "Confusion Matrix": conf_matrix
    }

    print(f"{name} Accuracy: {accuracy:.4f}")
    print(classification_report(y_test, predictions, target_names=["Negative", "Positive"]))
    print("-" * 50)
```

```
Random Forest Accuracy: 0.9290
              precision    recall  f1-score   support

    Negative       0.95      0.97      0.96     18292
    Positive       0.62      0.44      0.51      1708

    accuracy                           0.93     20000
   macro avg       0.78      0.71      0.74     20000
weighted avg       0.92      0.93      0.92     20000

--------------------------------------------------
K-Nearest Neighbors Accuracy: 0.9431
              precision    recall  f1-score   support

    Negative       0.95      0.99      0.97     18292
    Positive       0.85      0.41      0.55      1708

    accuracy                           0.94     20000
   macro avg       0.90      0.70      0.76     20000
weighted avg       0.94      0.94      0.93     20000

--------------------------------------------------
AdaBoost Classifier Accuracy: 0.9473
              precision    recall  f1-score   support

    Negative       0.95      1.00      0.97     18292
    Positive       1.00      0.38      0.55      1708

    accuracy                           0.95     20000
   macro avg       0.97      0.69      0.76     20000
weighted avg       0.95      0.95      0.94     20000

--------------------------------------------------
Decision Tree Classifier Accuracy: 0.9241
              precision    recall  f1-score   support

    Negative       0.95      0.97      0.96     18292
    Positive       0.57      0.47      0.51      1708

    accuracy                           0.92     20000
   macro avg       0.76      0.72      0.74     20000
weighted avg       0.92      0.92      0.92     20000

--------------------------------------------------
Logistic Regression Accuracy: 0.9401
              precision    recall  f1-score   support

    Negative       0.94      1.00      0.97     18292
    Positive       0.87      0.35      0.50      1708

    accuracy                           0.94     20000
   macro avg       0.91      0.67      0.73     20000
weighted avg       0.94      0.94      0.93     20000

--------------------------------------------------
```

In [208…

```
# Plot the confusion matrices
fig, axes = plt.subplots(2, 2, figsize=(14, 14))
fig.suptitle('Confusion Matrices of Different Models', fontsize=20)

for (name, result), ax in zip(results.items(), axes.flatten()):
    sns.heatmap(result['Confusion Matrix'], annot=True, fmt='d', cmap='Blues', ax=ax)
    ax.set_title(f'{name} (Accuracy: {result["Accuracy"]:.4f})')
    ax.set_xlabel('Predicted')
    ax.set_ylabel('Actual')

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



In [ ]:

Take New dataset for testing the models

In [209…

```
new_sample_data = pd.DataFrame({
    'smoking_history': np.random.choice([0, 1, 2], 10),  # 0: never, 1: current, 2: former
    'gender': np.random.choice([0, 1], 10),  # 0: Male, 1: Female
    'bmi': np.random.uniform(18, 40, 10),
    'blood_glucose_level': np.random.uniform(70, 200, 10)
})
```

In [210…

```
new_sample_data
```

Out[210]:

	smoking_history	gender	bmi	blood_glucose_level
0	0	1	26.604753	70.075834
1	0	1	31.469352	141.021477
2	2	1	18.445357	82.479776
3	0	0	18.130357	156.660308
4	2	1	24.560052	151.634666
5	1	0	38.117731	140.558481
6	1	0	30.616946	181.308104
7	1	1	29.731554	117.696247
8	2	0	19.389524	166.858844
9	0	0	22.620052	71.649262

In [211…
```
# Scale the new sample data
new_synthetic_data_scaled = scaler.transform(new_sample_data)
```

In [212…
```
# Predictions on new sample data
new_predictions = {}
for name, model in models:
    new_predictions[name] = model.predict(new_synthetic_data_scaled)
    print(f"{name} Predictions on New Data:", new_predictions[name])
```

```
Random Forest Predictions on New Data: [0 0 0 0 1 0 0 0 0 0]
K-Nearest Neighbors Predictions on New Data: [0 0 0 0 0 0 0 0 0 0]
AdaBoost Classifier Predictions on New Data: [0 0 0 0 0 0 0 0 0 0]
Decision Tree Classifier Predictions on New Data: [0 0 0 0 0 0 1 0 0 0]
Logistic Regression Predictions on New Data: [0 0 0 0 0 0 0 0 0 0]
```

In [ ]:

Important Note:-

Our problem statement specifies one important algorithm to use for solving this particular problem, which is linear regression. However, we are not using this algorithm. In our problem, where the target is to predict whether a person has diabetes (a binary classification problem), logistic regression is more appropriate than linear regression. That is why we use this linear regression algorithm.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: