Data preprocessing

These are the factors
A: Sex (0:male,1:female)
B: age
C: hypertension (0:doesn't have,1:has)
D: heart_disease (0:doesn't have,1:has)
E: ever_married (0:no,1:yes)
F: work_type is ignored (0:"children", 1:"Govtjov", 2:"Never worked", 3:"Private", 4:"Self-employed)
G: Residence_type (0:Rural,1:Urban)
H: avg_glucose_level
I: BMI
J: smoking_status (0:never smoked, 1:formerly smoked, 2:smokes) "unknown" is ignored
K: stroke(0:no stroke,1:has a stroke)

In [127…
```
import pandas as pd
dataset = pd.read_csv("stroke_data2.csv")
dataset = dataset.sample(frac=1)
dataset.reset_index(drop=True, inplace=True)
dataset
```

Out[127]:

	sex	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi
0	1	79	1	0	1	4	1	221.79	30.0
1	1	47	0	0	0	4	0	154.08	20.2
2	1	63	0	0	1	4	1	93.58	25.1
3	1	31	0	0	0	3	0	77.52	21.9
4	1	9	0	0	1	2	1	64.99	33.2
...	...	...	...	...	...	...	...	...	...
40844	0	56	0	1	1	4	1	91.92	35.9
40845	0	49	0	0	1	4	0	65.47	24.1
40846	1	25	0	0	1	4	0	100.98	28.2
40847	0	28	0	0	1	4	1	86.94	41.1
40848	1	42	0	0	1	4	0	95.36	25.1

40849 rows × 11 columns

There are 40849 rows of patients' data. After shuffling the rows, we choose the first 40000 rows as the training data, and 800 rows as validation data, and the last 10 rows as testing data.

In [128…
```
from sklearn.preprocessing import StandardScaler

x = dataset.iloc[:,:-1]
y= dataset.iloc[:,-1]
x_train = x.iloc[:40000]
y_train = y.iloc[:40000]
x_valid = x.iloc[40039:40839]
```

```
y_valid = y.iloc[40039:40839]
x_test = x.iloc[40839:]
y_test = y.iloc[40839:]

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_valid = scaler.transform(x_valid)
x_test = scaler.transform(x_test)
```

Logistic Regression

In [129…
```
from sklearn.linear_model import LogisticRegression
model_LR = LogisticRegression(penalty='l2', C=0.0001,solver="lbfgs")
model_LR.fit(x_train,y_train)
```

Out[129]:
```
LogisticRegression(C=0.0001)
```

Multilayer Perceptron

In [130…
```
from sklearn.neural_network import MLPClassifier
model_MLP = MLPClassifier(hidden_layer_sizes=(100, 100), activation='tanh', solver='lbfg
model_MLP.fit(x_train, y_train)
```

Out[130]:
```
MLPClassifier(activation='tanh', hidden_layer_sizes=(100, 100), max_iter=1000,
              random_state=42, solver='lbfgs')
```

DecisionTreeClassifier

In [131…
```
from sklearn.tree import DecisionTreeClassifier
model_DT = DecisionTreeClassifier(splitter = "best", random_state = 42)
model_DT.fit(x_train, y_train)
```

Out[131]:
```
DecisionTreeClassifier(random_state=42)
```

Random Forest

In [132…
```
from sklearn.ensemble import RandomForestClassifier
model_RF = RandomForestClassifier(bootstrap=True, random_state = 42)
model_RF.fit(x_train, y_train)
```

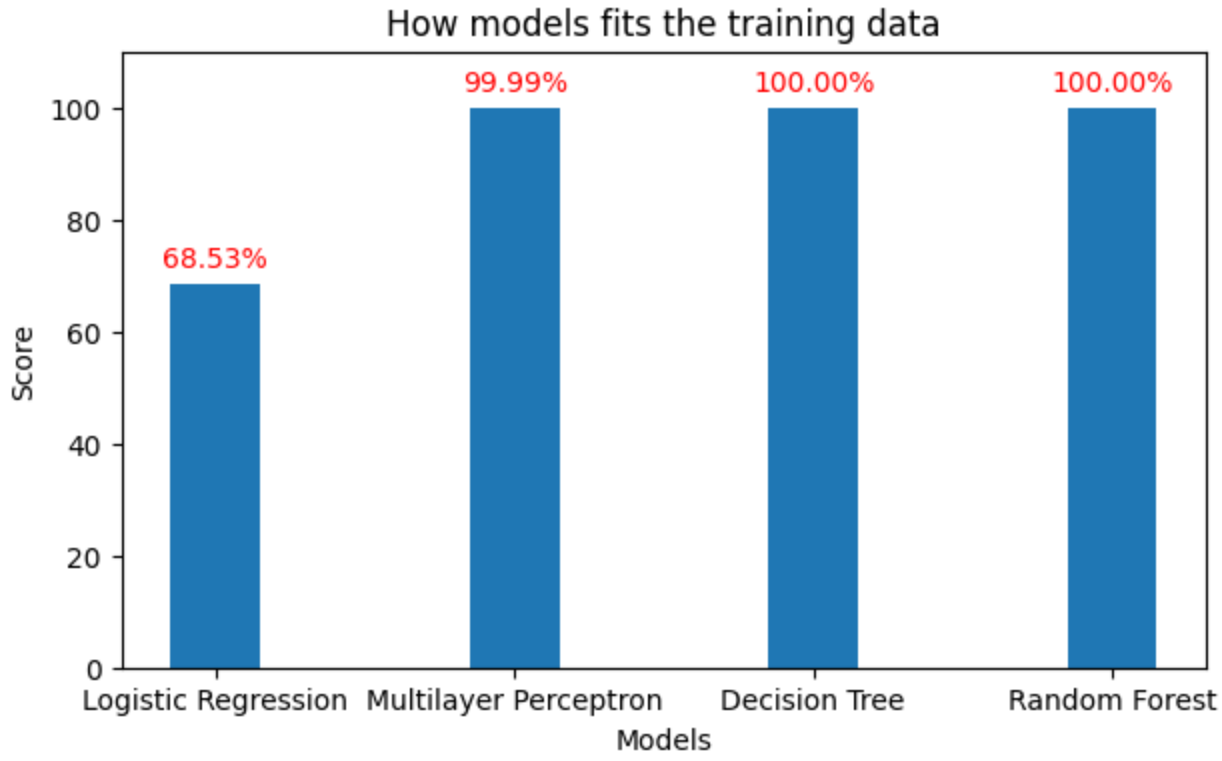
Out[132]:
```
RandomForestClassifier(random_state=42)
```

Evaluation

How models fits the training data

In [182…

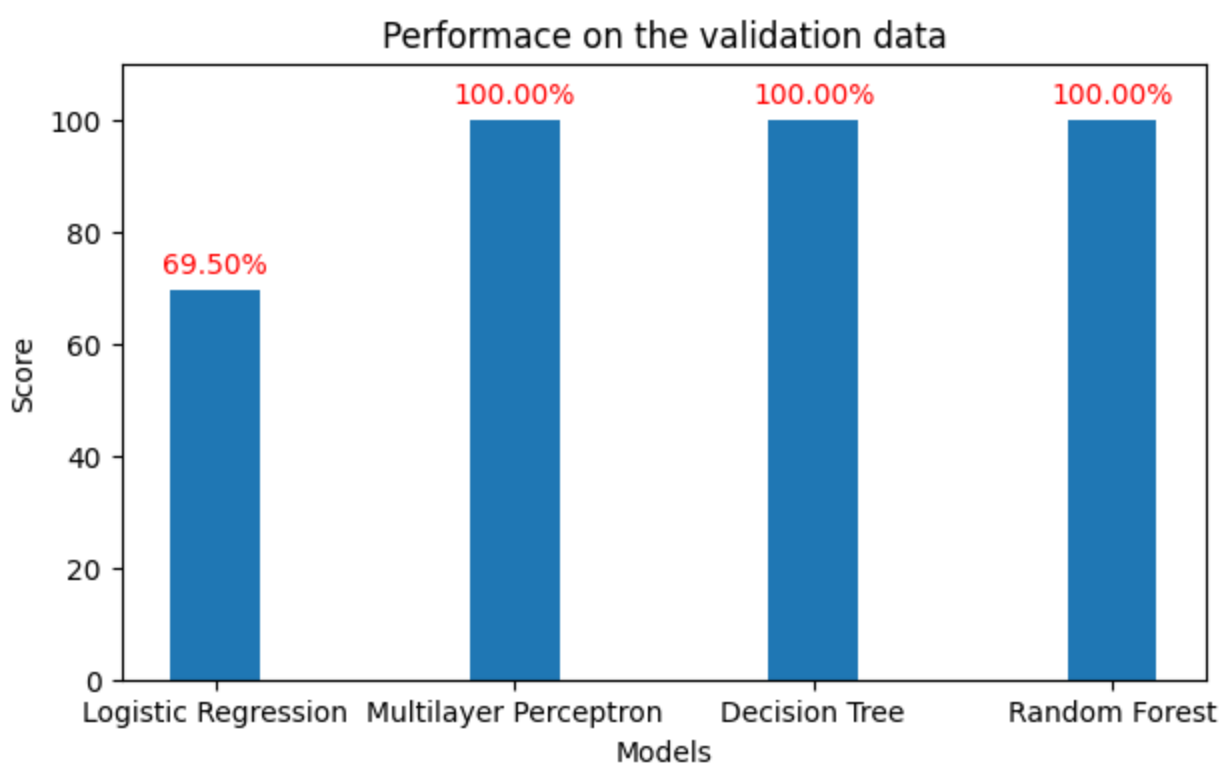
```
import matplotlib.pyplot as plt
import numpy as np
labels = ["Logistic Regression", "Multilayer Perceptron", "Decision Tree", "Random Fores
trainScore = np.array([model_LR.score(x_train,y_train),model_MLP.score(x_train,y_train),
trainScore = trainScore*100
size = len(labels)
plt.figure(figsize=(7, 4))
for i in range(size):
    plt.text(i,trainScore[i]+3,f'{trainScore[i]:.2f}%',ha = 'center',color='red')
plt.xlabel('Models')
plt.ylabel('Score')
plt.title('How models fits the training data')
plt.bar(range(size),trainScore,width=0.3)
plt.xticks(range(size), labels)
plt.ylim(0,110)
plt.show()
```



The performance of models on the validation data

In [183…

```
trainScore = np.array([model_LR.score(x_valid,y_valid),model_MLP.score(x_valid,y_valid),
trainScore = trainScore*100
size = len(labels)
plt.figure(figsize=(7, 4))
for i in range(size):
    plt.text(i,trainScore[i]+3,f'{trainScore[i]:.2f}%',ha = 'center',color='red')
plt.xlabel('Models')
plt.ylabel('Score')
plt.title('Performace on the validation data')
plt.bar(range(size),trainScore,width=0.3)
plt.xticks(range(size), labels)
plt.ylim(0,110)
plt.show()
```



We conclude that random forest classifier model works the best. Therefore, we choose random forest model for our final testing.

Final testing

In [260…

```
y_final_pred = pd.Series(model_MLP.predict(x_test))
y_final_prob = pd.DataFrame(model_MLP.predict_proba(x_test)).iloc[:,1]
leng = len(y_final_pred)
cnt = 0

for i in range(leng):
    print(f"Patient #{i+1}: Probability of having stroke is {y_final_prob.iloc[i]*100:.2
    if y_final_pred.iloc[i] == 0:
        print("predict that do not have a stroke.",end=" ")
    elif y_final_pred.iloc[i] == 1:
        print("predict that have a stroke.",end=" ")
    if y_final_pred.iloc[i]==y_test.iloc[i]:
        cnt+=10
        print('\033[32m'+"We predicted correctly."+'\033[0m')
    else:
        print('\033[31m'+"We didn't predict correctly."+'\033[0m')

plt.figure(figsize=(3, 3))
plt.pie([cnt, 100 - cnt], labels=[f'Correctly', f'Non-Correctly'],colors=['blue', 'red']
plt.title('Accuracy of Prediction')
plt.legend([f'correctly: {cnt}%', f'incorrectly{100 - cnt}%'],bbox_to_anchor=(0.7,0.3))
plt.show()
```

```
Patient #1: Probability of having stroke is 0.00%, predict that do not have a stroke. We
 predicted correctly.
Patient #2: Probability of having stroke is 0.04%, predict that do not have a stroke. We
 predicted correctly.
Patient #3: Probability of having stroke is 100.00%, predict that have a stroke. We pred
icted correctly.
Patient #4: Probability of having stroke is 0.00%, predict that do not have a stroke. We
 predicted correctly.
Patient #5: Probability of having stroke is 0.00%, predict that do not have a stroke. We
```

predicted correctly.
Patient #6: Probability of having stroke is 100.00%, predict that have a stroke. We predicted correctly.
Patient #7: Probability of having stroke is 0.00%, predict that do not have a stroke. We predicted correctly.
Patient #8: Probability of having stroke is 100.00%, predict that have a stroke. We predicted correctly.
Patient #9: Probability of having stroke is 100.00%, predict that have a stroke. We predicted correctly.
Patient #10: Probability of having stroke is 0.00%, predict that do not have a stroke. We predicted correctly.