In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [ ]:
```
import os
for dirname, _, filenames in os.walk('/content/brain_stroke.csv'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [ ]:
```
data=pd.read_csv('/content/brain_stroke.csv')
```

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

Out [4]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	str
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4981 entries, 0 to 4980
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             4981 non-null   object 
 1   age                4981 non-null   float64
 2   hypertension       4981 non-null   int64  
 3   heart_disease      4981 non-null   int64  
 4   ever_married       4981 non-null   object 
 5   work_type          4981 non-null   object 
 6   Residence_type     4981 non-null   object 
 7   avg_glucose_level  4981 non-null   float64
 8   bmi                4981 non-null   float64
 9   smoking_status     4981 non-null   object 
 10  stroke             4981 non-null   int64  
dtypes: float64(3), int64(3), object(5)
memory usage: 428.2+ KB
```

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

Out [6]:
```
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                  0
smoking_status       0
stroke               0
dtype: int64
```

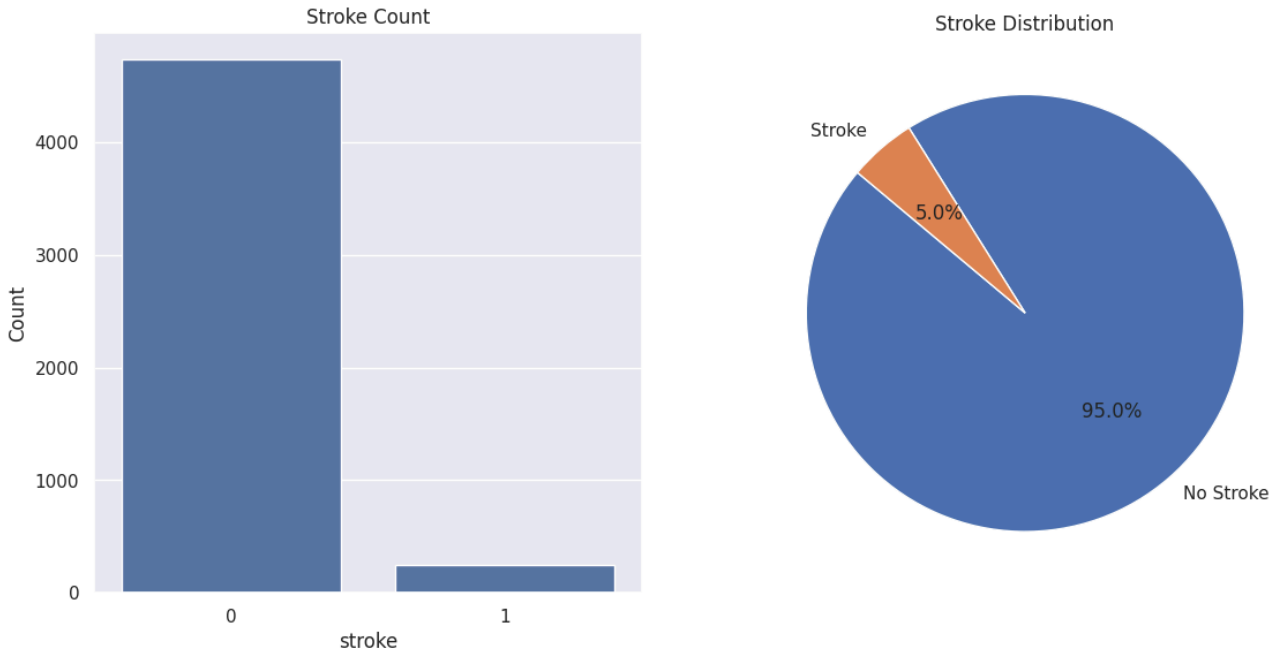
In [ ]:
```
data = data.drop(["ever_married", "work_type", "Residence_type"], axis =1)
```

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

Out [8]:

	gender	age	hypertension	heart_disease	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	105.92	32.5	never smoked	1
2	Female	49.0	0	0	171.23	34.4	smokes	1
3	Female	79.0	1	0	174.12	24.0	never smoked	1
4	Male	81.0	0	0	186.21	29.0	formerly smoked	1

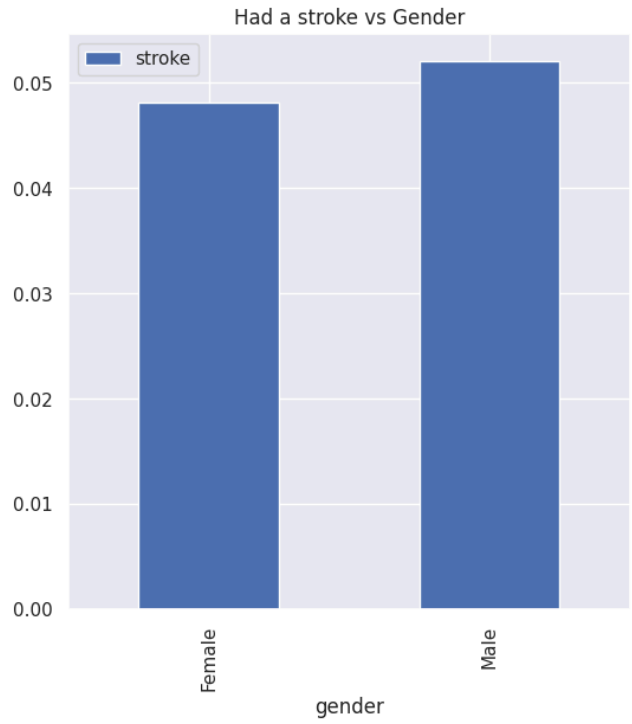
In [ ]:
```
sns.set(style="darkgrid")
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
sns.countplot(x='stroke', data=data, ax=ax[0])
ax[0].set_title('Stroke Count')
ax[0].set_ylabel('Count')
labels = ['No Stroke', 'Stroke']
sizes = data['stroke'].value_counts()
ax[1].pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
ax[1].set_title('Stroke Distribution')

plt.show()
```

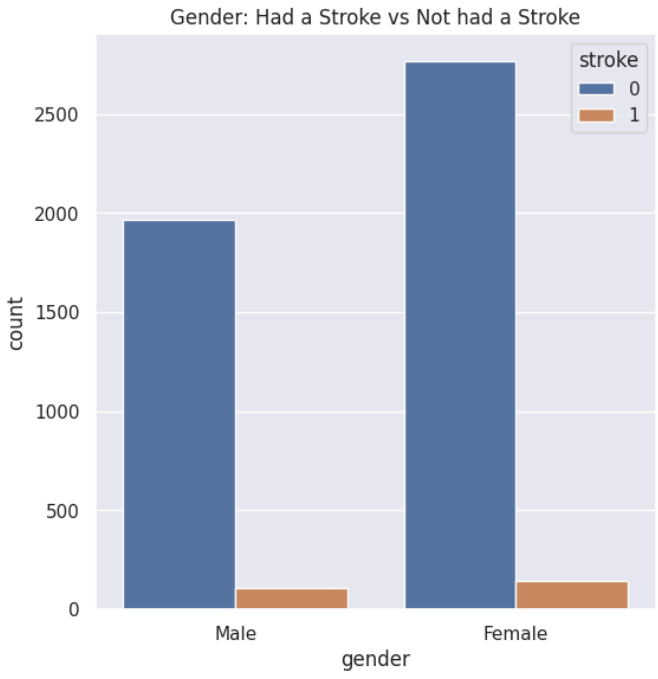


In [ ]:
```
data.groupby(['gender','stroke'])['stroke'].count()
```

Out [10]:
```
gender  stroke
Female  0         2767
        1          140
Male    0         1966
        1          108
Name: stroke, dtype: int64
```

In [ ]:
```
sns.set(style="darkgrid")
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
data[['gender', 'stroke']].groupby(['gender']).mean().plot.bar(ax=ax[0])
ax[0].set_title('Had a stroke vs Gender')
sns.countplot(x='gender', hue='stroke', data=data, ax=ax[1])
ax[1].set_title('Gender: Had a Stroke vs Not had a Stroke')
plt.show()
```

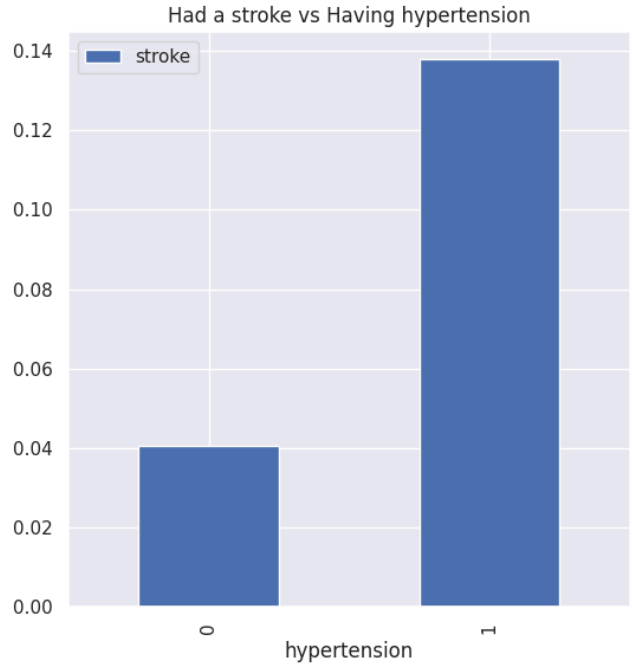


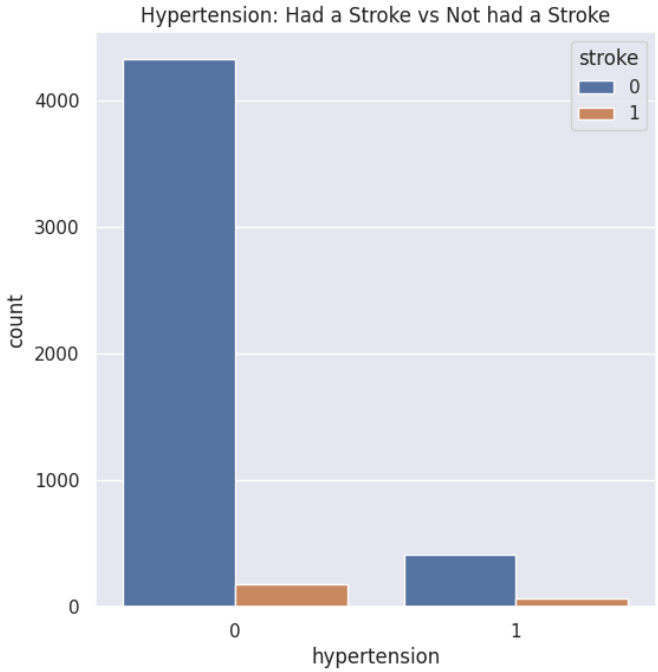


In [ ]:
```
data.groupby(['hypertension','stroke'])['stroke'].count()
```

Out [12]:
```
hypertension  stroke
0             0         4320
              1          182
1             0          413
              1           66
Name: stroke, dtype: int64
```

In [ ]:
```
sns.set(style="darkgrid")
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
data[['hypertension', 'stroke']].groupby(['hypertension']).mean().plot.bar(ax=ax[0])
ax[0].set_title('Had a stroke vs Having hypertension')
sns.countplot(x='hypertension', hue='stroke', data=data, ax=ax[1])
ax[1].set_title('Hypertension: Had a Stroke vs Not had a Stroke')
plt.show()
```





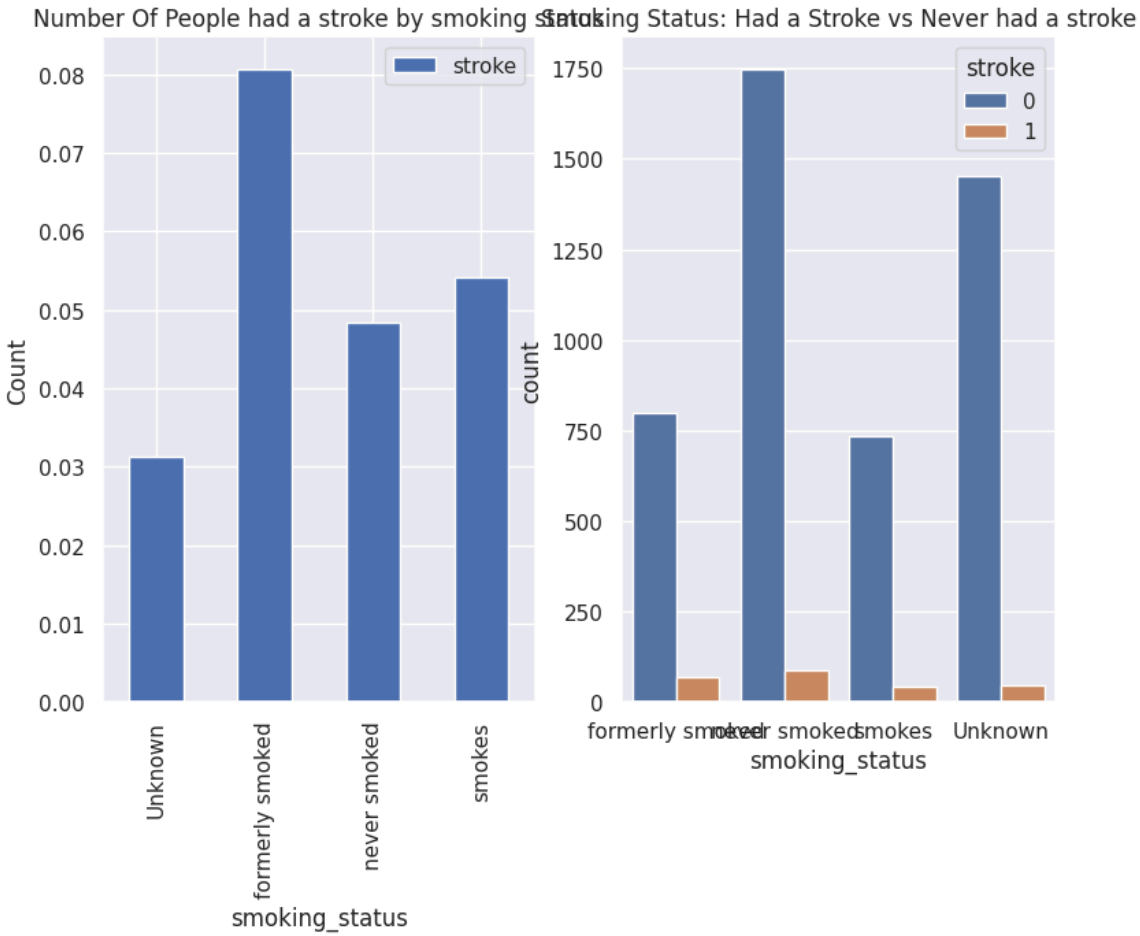
In [ ]:
```
pd.crosstab(data.smoking_status,data.stroke,margins=True).style.background_gradient(cmap='summer_r')
```

Out [14]:

stroke	0	1	All
smoking_status			
Unknown	1453	47	1500
formerly smoked	797	70	867
never smoked	1749	89	1838

stroke	0	1	All
smoking_status			
smokes	734	42	776
All	4733	248	4981

In [ ]:
```
sns.set(style="darkgrid")
fig, ax = plt.subplots(1, 2, figsize=(8, 6))
data[['smoking_status', 'stroke']].groupby(['smoking_status']).mean().plot.bar(ax=ax[0])
ax[0].set_title('Number Of People had a stroke by smoking status')
ax[0].set_ylabel('Count')
sns.countplot(x='smoking_status', hue='stroke', data=data, ax=ax[1])
ax[1].set_title('Smoking Status: Had a Stroke vs Never had a stroke')
plt.show()
```

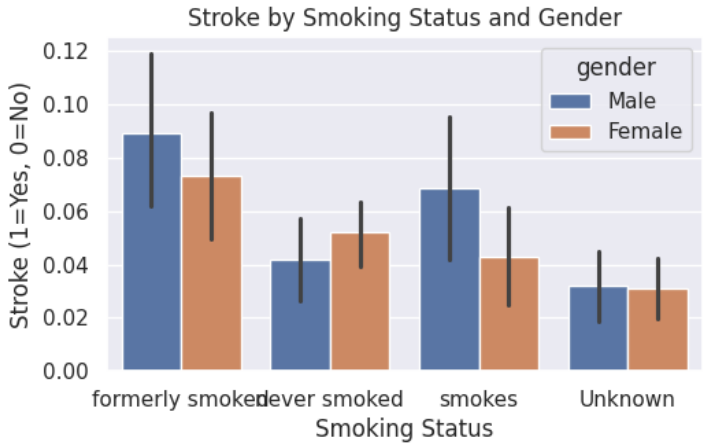


In [ ]:
```
pd.crosstab([data.gender,data.stroke],data.smoking_status,margins=True).style.background_gradient(cm
```

Out [16]:

gender	stroke	smoking_status	Unknown	formerly smoked	never smoked	smokes	All
Female	0		783	430	1132	422	2767
	1		25	34	62	19	140
Male	0		670	367	617	312	1966
	1		22	36	27	23	108
All			1500	867	1838	776	4981

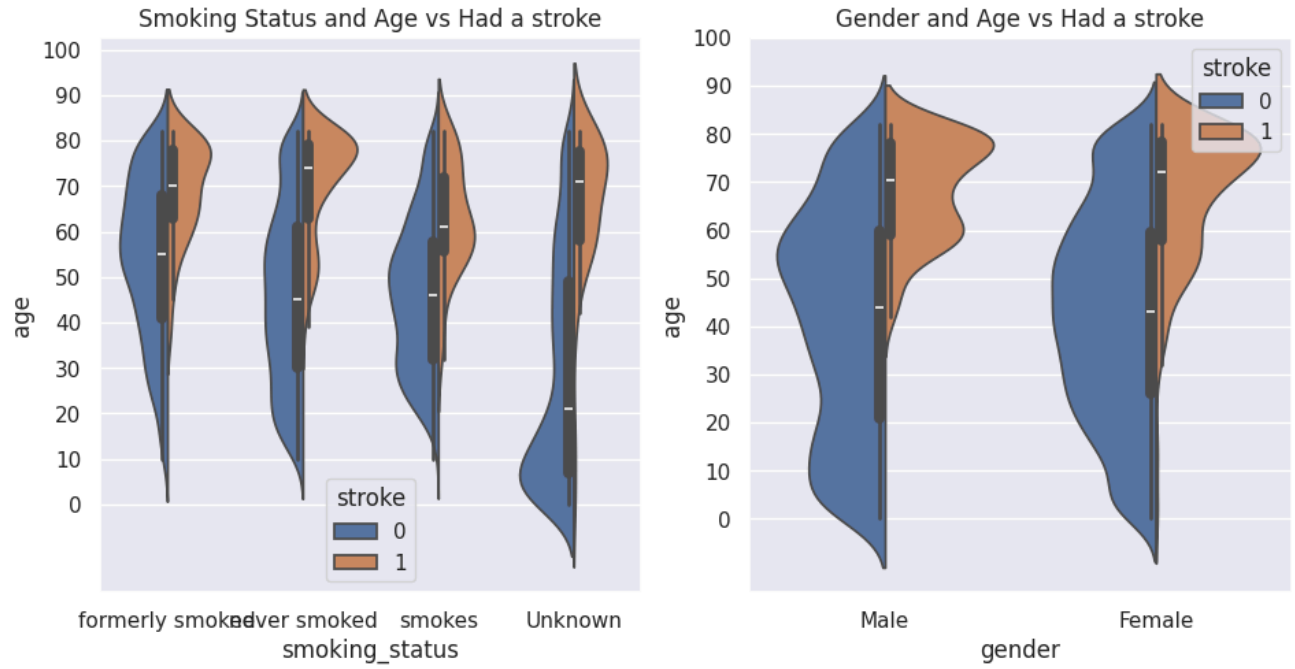
In [ ]:
```
plt.figure(figsize=(5, 3))
sns.barplot(x='smoking_status', y='stroke', hue='gender', data=data)
plt.title('Stroke by Smoking Status and Gender')
plt.xlabel('Smoking Status')
plt.ylabel('Stroke (1=Yes, 0=No)')
plt.show()
```



In [ ]:
```
print('Oldest Person was of:',data['age'].max(),'Years')
print('Youngest Person was of:',data['age'].min(),'Years')
print('Average Age in the data:',data['age'].mean(),'Years')
```

```
Oldest Person was of: 82.0 Years
Youngest Person was of: 0.08 Years
Average Age in the data: 43.41985946597069 Years
```

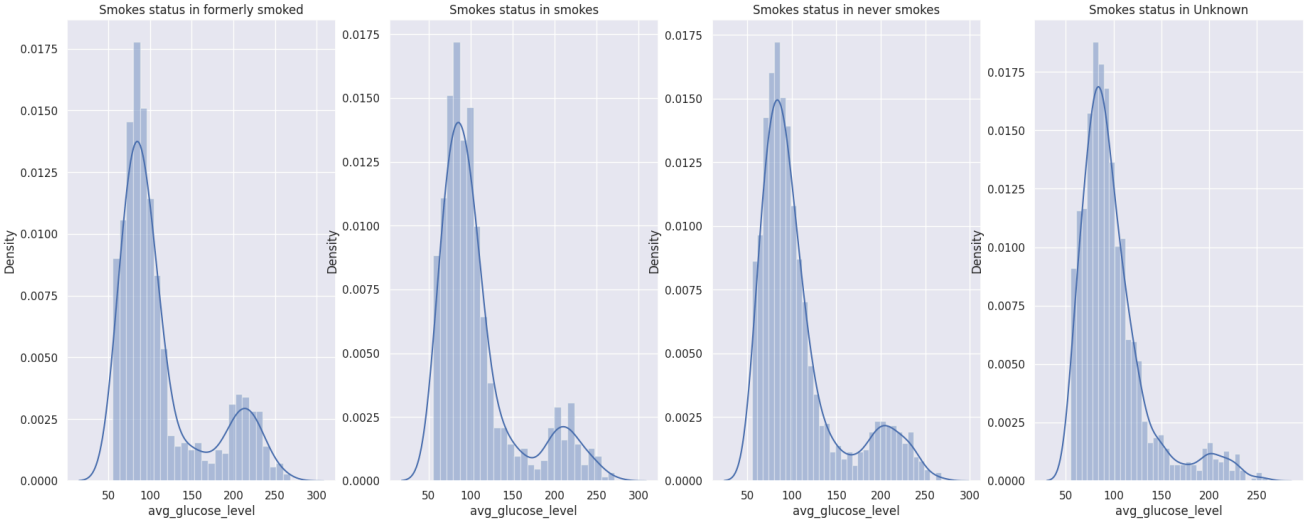
In [ ]:
```
f, ax = plt.subplots(1, 2, figsize=(10, 5))
sns.violinplot(x="smoking_status", y="age", hue="stroke", data=data, split=True, ax=ax[0])
ax[0].set_title('Smoking Status and Age vs Had a stroke')
ax[0].set_yticks(range(0, 110, 10))

sns.violinplot(x="gender", y="age", hue="stroke", data=data, split=True, ax=ax[1])
ax[1].set_title('Gender and Age vs Had a stroke')
ax[1].set_yticks(range(0, 110, 10))

plt.show()
```



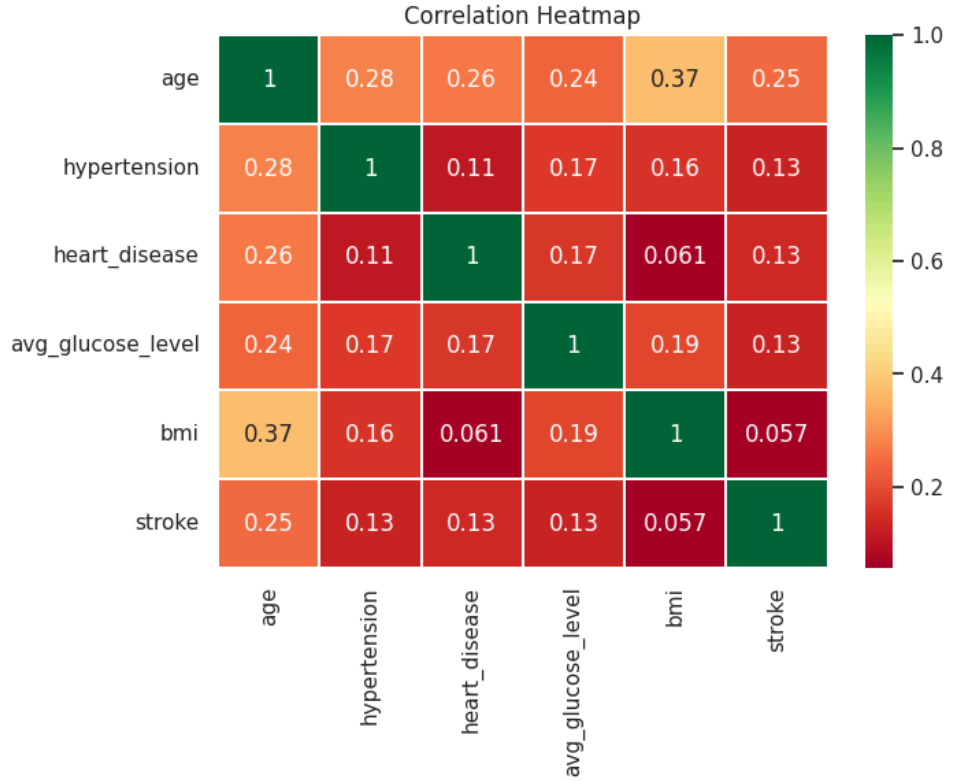
In [ ]:
```
print('Maximum "Average Glucose Level" in data:',data['avg_glucose_level'].max())
print('Minimum "Average Glucose Level" in data:',data['avg_glucose_level'].min())
print('Average "Average Glucose Level" in data:',data['avg_glucose_level'].mean())
print("**************************************")
print('Maximum "BMI" in data:',data['bmi'].max())
print('Minimum "BMI" in data:',data['bmi'].min())
print('Average "BMI" in data:',data['bmi'].mean())
```

```
Maximum "Average Glucose Level" in data: 271.74
Minimum "Average Glucose Level" in data: 55.12
Average "Average Glucose Level" in data: 105.94356153382854
**************************************
Maximum "BMI" in data: 48.9
Minimum "BMI" in data: 14.0
Average "BMI" in data: 28.498173057618956
```

In [ ]:
```
f,ax=plt.subplots(1,4,figsize=(20,8))
sns.distplot(data[data['smoking_status']=="formerly smoked"].avg_glucose_level,ax=ax[0])
ax[0].set_title('Smokes status in formerly smoked')
sns.distplot(data[data['smoking_status']=="smokes"].avg_glucose_level,ax=ax[1])
ax[1].set_title('Smokes status in smokes')
sns.distplot(data[data['smoking_status']== "never smoked"].avg_glucose_level,ax=ax[2])
ax[2].set_title('Smokes status in never smokes')
sns.distplot(data[data['smoking_status']== "Unknown"].avg_glucose_level,ax=ax[3])
ax[3].set_title('Smokes status in Unknown')
plt.show()
```



In [ ]:
```
numeric_data = data.select_dtypes(include=['number'])
sns.heatmap(numeric_data.corr(), annot=True, cmap='RdYlGn', linewidths=0.2)
plt.title('Correlation Heatmap')
plt.figure(figsize=(5, 3))
plt.show()
```




```
<Figure size 500x300 with 0 Axes>
```

In [ ]:
```
data.insert(2,'age_band', np.zeros)
```

In [ ]:
```
data.loc[data['age']<=16,'age_band']=0
data.loc[(data['age']>16)&(data['age']<=32),'age_band']=1
data.loc[(data['age']>32)&(data['age']<=48),'age_band']=2
data.loc[(data['age']>48)&(data['age']<=64),'age_band']=3
data.loc[data['age']>64,'age_band']=4
data.drop(columns= "age", inplace=True)
```

```
data["age_band"]=data['age_band'].astype(str).astype(int)
data.head()
```

Out [24]:

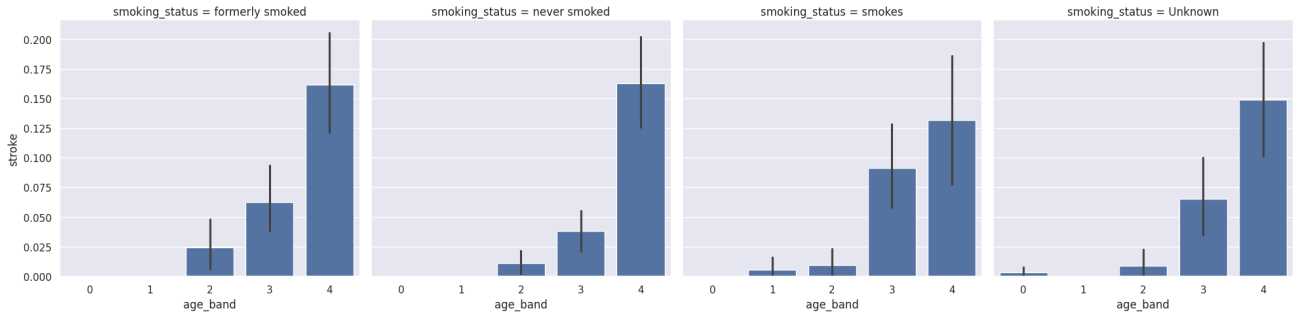
	gender	age_band	hypertension	heart_disease	avg_glucose_level	bmi	smoking_status	stroke
0	Male	4	0	1	228.69	36.6	formerly smoked	1
1	Male	4	0	1	105.92	32.5	never smoked	1
2	Female	3	0	0	171.23	34.4	smokes	1
3	Female	4	1	0	174.12	24.0	never smoked	1
4	Male	4	0	0	186.21	29.0	formerly smoked	1

In [ ]:
```
data['age_band'].value_counts().to_frame().style.background_gradient(cmap='summer')
```

Out [25]:

age_band	count
3	1229
2	1067
4	1020
1	895
0	770

In [ ]:
```
sns.set_context("notebook")
sns.catplot(x='age_band', y='stroke', data=data, col='smoking_status', kind='bar')
plt.show()
```



In [ ]:
```
data = data.copy()
column = 'avg_glucose_level'
column2 = 'bmi'
data[column] = data[column] /data[column].abs().max()
data[column2] = data[column2] /data[column2].abs().max()
display(data[column], data[column2])
```

```
0       0.841577
1       0.389784
2       0.630124
3       0.640760
4       0.685251
          ...
4976    0.258151
4977    0.703430
4978    0.349672
4979    0.308898
4980    0.308199
Name: avg_glucose_level, Length: 4981, dtype: float64
0       0.748466
1       0.664622
2       0.703476
3       0.490798
4       0.593047
          ...
4976    0.609407
4977    0.635992
4978    0.650307
4979    0.613497
4980    0.595092
Name: bmi, Length: 4981, dtype: float64
```

In [ ]:
```
data['gender'].replace(['Male','Female'],[0,1],inplace=True)
data['gender'].head()
```

Out [28]:
```
0    0
1    0
2    1
3    1
4    0
Name: gender, dtype: int64
```

In [ ]:
```
data["smoking_status"].unique()
```

Out [29]:
```
array(['formerly smoked', 'never smoked', 'smokes', 'Unknown'],
      dtype=object)
```

In [ ]:
```
data['smoking_status'].replace(['formerly smoked', 'never smoked', 'smokes', 'Unknown'],[0,1,2,3],in
data['smoking_status'].head()
```

Out [30]:
```
0    0
1    1
2    2
3    1
4    0
Name: smoking_status, dtype: int64
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression #logistic regression
from sklearn import svm #support vector Machine
from sklearn.ensemble import RandomForestClassifier #Random Forest
from sklearn.neighbors import KNeighborsClassifier #KNN
from sklearn.naive_bayes import GaussianNB #Naive bayes
from sklearn.tree import DecisionTreeClassifier #Decision Tree
from sklearn.model_selection import train_test_split #training and testing data split
from sklearn import metrics #accuracy measure
from sklearn.metrics import confusion_matrix #for confusion matrix
```

In [ ]:
```
train,test=train_test_split(data,test_size=0.3,random_state=0,stratify=data['stroke'])
train_X=train[train.columns[:-1]]
train_Y=train[train.columns[-1:]]
test_X=test[test.columns[:-1]]
test_Y=test[test.columns[-1:]]
X=data[data.columns[:-1]]
Y=data["stroke"]
len(train_X), len(train_Y), len(test_X), len(test_Y)
```

Out [32]:
```
(3486, 3486, 1495, 1495)
```

In [ ]:
```
print(train_X)
```

```
      gender  age_band  hypertension  heart_disease  avg_glucose_level  \
4103       0         4             0              0           0.316405
1999       1         0             0              0           0.303194
1074       1         4             0              0           0.250092
1264       1         3             0              0           0.325348
2914       1         3             0              0           0.329690
...      ...       ...           ...            ...                ...
1522       1         3             0              1           0.765622
2808       1         0             0              0           0.301465
4547       1         0             0              0           0.270406
4593       1         2             0              0           0.254140
1512       1         2             0              0           0.378634

          bmi  smoking_status
4103  0.572597               3
1999  0.349693               1
1074  0.548057               3
1264  0.517382               0
2914  0.562372               3
...        ...             ...
1522  0.721881               2
2808  0.464213               3
4547  0.468303               3
4593  0.593047               2
1512  0.760736               0

[3486 rows x 7 columns]
```

In [ ]:
```
print(train_Y)
```

```
      stroke
4103       0
1999       0
1074       0
1264       0
2914       0
...      ...
1522       0
2808       0
4547       0
4593       0
1512       0

[3486 rows x 1 columns]
```

In [ ]:
```
#svm
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3)
```

```
model=SVC(kernel='linear')
model.fit(X_train,Y_train)
y_pred=model.predict(X_test)
from sklearn.metrics import accuracy_score
print("Linear SVM",accuracy_score(Y_test,y_pred))
```

```
Linear SVM 0.9464882943143813
```

In [ ]:
```
from sklearn.metrics import classification_report
print(classification_report(Y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      1.00      0.97      1415
           1       0.00      0.00      0.00        80

    accuracy                           0.95      1495
   macro avg       0.47      0.50      0.49      1495
weighted avg       0.90      0.95      0.92      1495
```

In [ ]:
```
model = LogisticRegression()
model.fit(train_X,train_Y)
y_pred=model.predict(test_X)
print('The accuracy of the Logistic Regression is',metrics.accuracy_score(y_pred,test_Y))
```

```
The accuracy of the Logistic Regression is 0.9505016722408027
```

In [ ]:
```
model=KNeighborsClassifier()
model.fit(train_X,train_Y)
y_pred=model.predict(test_X)
print('The accuracy of the KNN is',metrics.accuracy_score(y_pred,test_Y))
```

```
The accuracy of the KNN is 0.9444816053511705
```

In [ ]:
```
#PERCEPTRON
from sklearn.linear_model import Perceptron
perceptron_model = Perceptron()
perceptron_model.fit(X_train, Y_train)
y_pred = perceptron_model.predict(X_test)
accuracy = accuracy_score(Y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9464882943143813
```

In [ ]:
```
model=DecisionTreeClassifier()
model.fit(train_X,train_Y)
prediction4=model.predict(test_X)
print('The accuracy of the Decision Tree is',metrics.accuracy_score(prediction4,test_Y))
```

```
The accuracy of the Decision Tree is 0.9103678929765886
```

In [ ]:
```
model=GaussianNB()
model.fit(train_X,train_Y)
prediction6=model.predict(test_X)
print('The accuracy of the NaiveBayes is',metrics.accuracy_score(prediction6,test_Y))
```

```
The accuracy of the NaiveBayes is 0.862876254180602
```

In [ ]:
```
model=RandomForestClassifier(n_estimators=100)
model.fit(train_X,train_Y)
prediction7=model.predict(test_X)
print('The accuracy of the Random Forests is',metrics.accuracy_score(prediction7,test_Y))
```

```
The accuracy of the Random Forests is 0.9444816053511705
```

In [ ]:
```
from sklearn.model_selection import KFold #for K-fold cross validation
from sklearn.model_selection import cross_val_score #score evaluation
from sklearn.model_selection import cross_val_predict #prediction
kfold = KFold(n_splits=10) # k=10, split the data into 10 equal parts
cv_mean=[]
accuracy=[]
#std=[]
classifiers=['Linear Svm','Perceptron','Logistic Regression','KNN','Decision Tree','Naive Bayes','Ra
models=[svm.SVC(kernel='linear'),Perceptron(),LogisticRegression(),KNeighborsClassifier(n_neighbors=
```

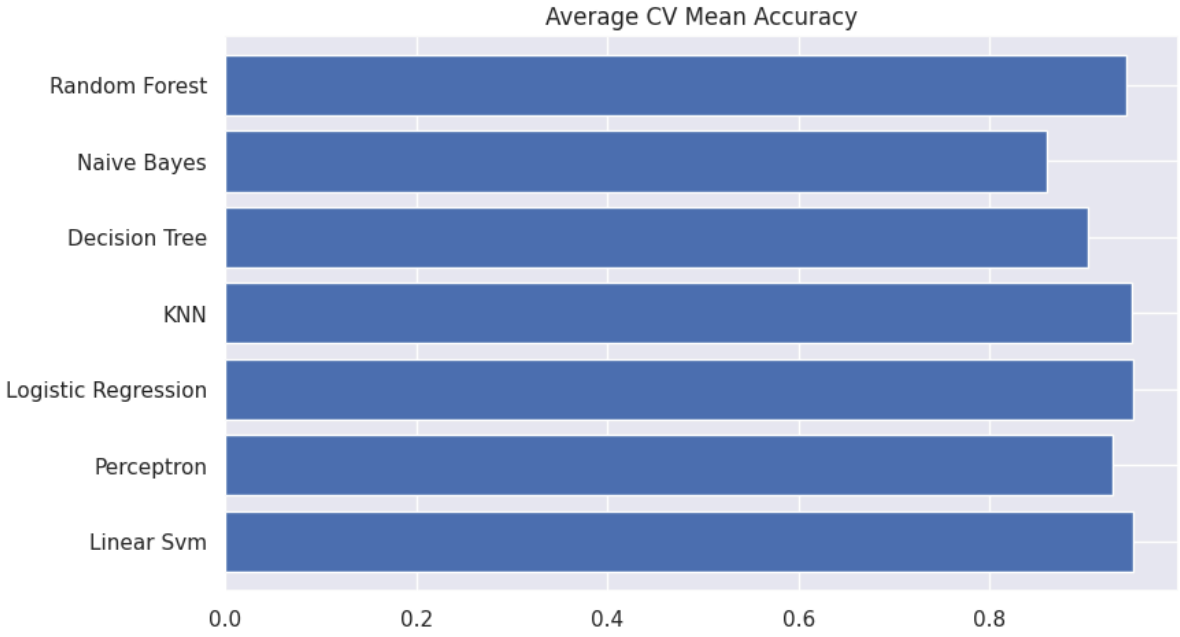
```
for i in models:
    model = i
    cv_result = cross_val_score(model,X,Y, cv = kfold,scoring = "accuracy")
    cv_result=cv_result
    cv_mean.append(cv_result.mean())
    #std.append(cv_result.std())
    accuracy.append(cv_result)
new_models_dataframe2=pd.DataFrame({'Accuracy':cv_mean},index=classifiers)
new_models_dataframe2
```

Out [43]:

	Accuracy
Linear Svm	0.950285
Perceptron	0.928799
Logistic Regression	0.950285
KNN	0.949080
Decision Tree	0.901890
Naive Bayes	0.859911
Random Forest	0.943056

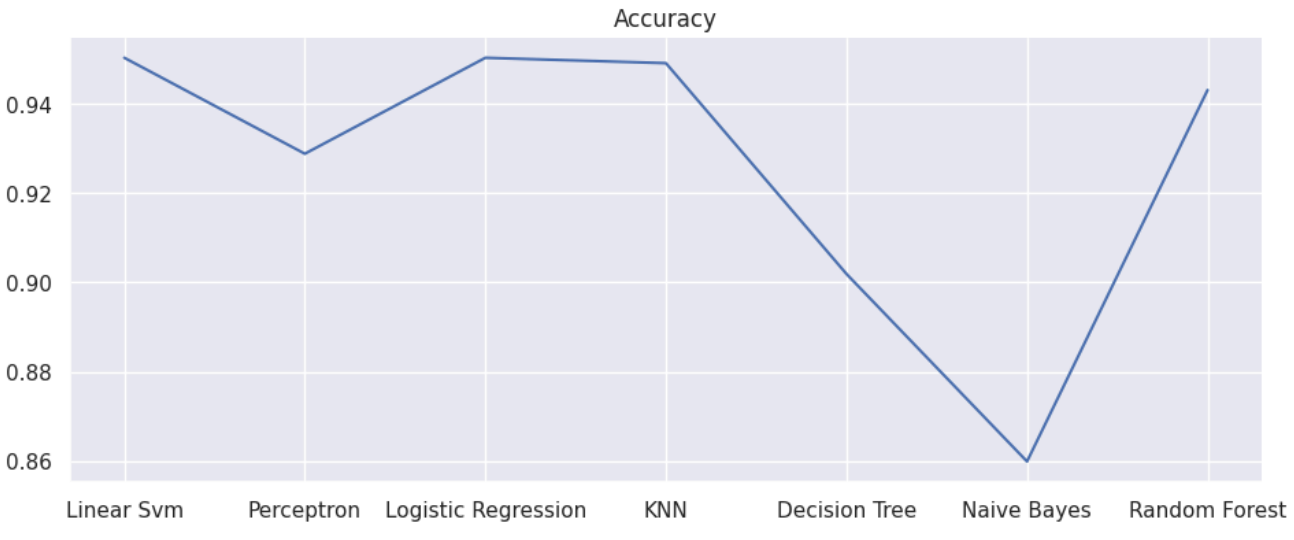
In [ ]:

```
new_models_dataframe2['Accuracy'].plot.barh(width=0.8)
plt.title('Average CV Mean Accuracy')
fig=plt.gcf()
fig.set_size_inches(8,5)
plt.show()
```

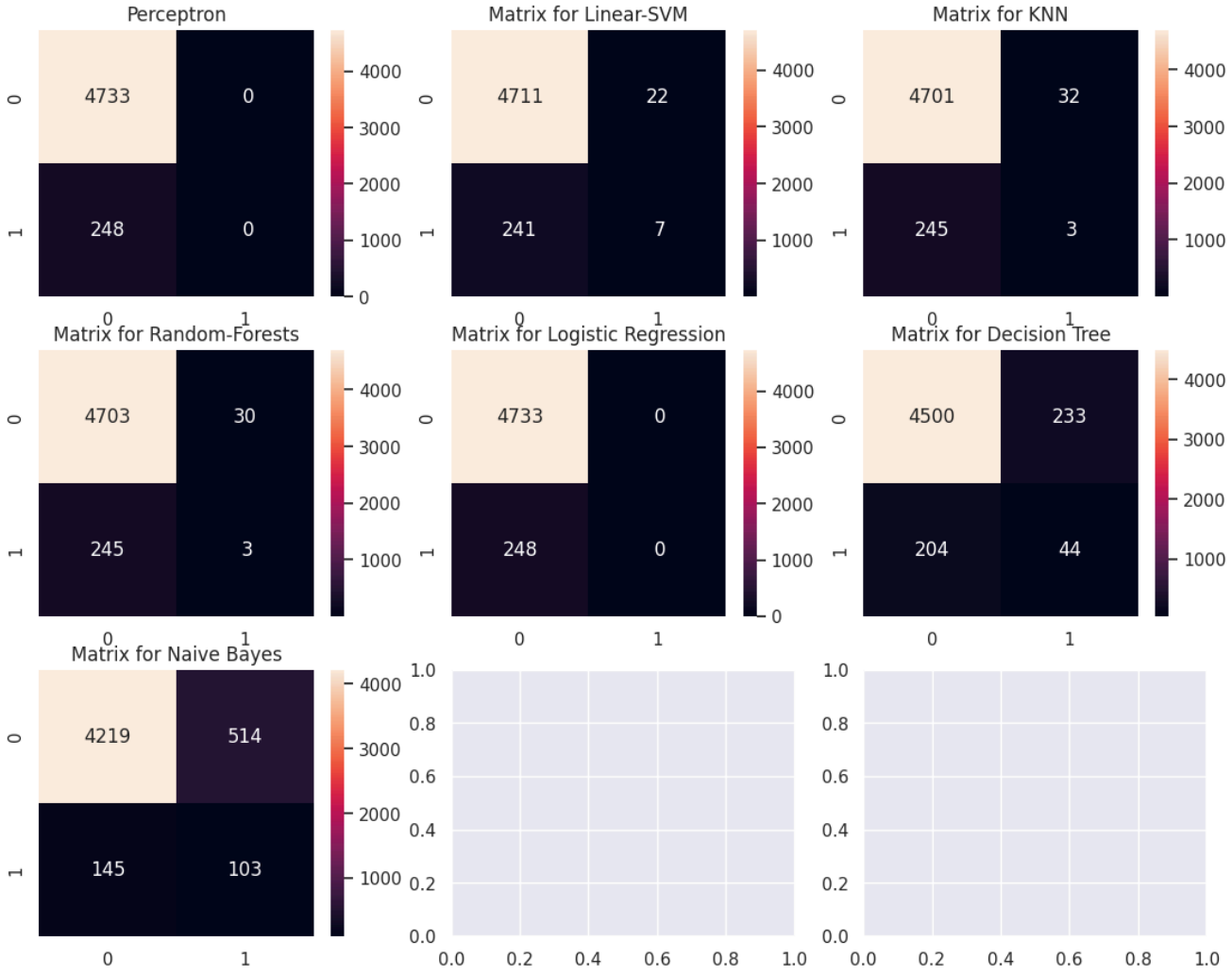


In [ ]:

```
from matplotlib import pyplot as plt
new_models_dataframe2['Accuracy'].plot(kind='line', figsize=(10, 4), title='Accuracy')
plt.gca().spines[['top', 'right']].set_visible(False)
```



In [ ]:
```
f,ax=plt.subplots(3,3,figsize=(12,10))
y_pred = cross_val_predict(svm.SVC(kernel='rbf'),X,Y,cv=10)
sns.heatmap(confusion_matrix(Y,y_pred),ax=ax[0,0],annot=True,fmt='2.0f')
ax[0,1].set_title('Matrix for Linear-SVM')
y_pred = cross_val_predict(KNeighborsClassifier(n_neighbors=5),X,Y,cv=10)
sns.heatmap(confusion_matrix(Y,y_pred),ax=ax[0,2],annot=True,fmt='2.0f')
ax[0,2].set_title('Matrix for KNN')
y_pred = cross_val_predict(RandomForestClassifier(n_estimators=100),X,Y,cv=10)
sns.heatmap(confusion_matrix(Y,y_pred),ax=ax[1,0],annot=True,fmt='2.0f')
ax[1,0].set_title('Matrix for Random-Forests')
y_pred = cross_val_predict(LogisticRegression(),X,Y,cv=10)
sns.heatmap(confusion_matrix(Y,y_pred),ax=ax[1,1],annot=True,fmt='2.0f')
ax[1,1].set_title('Matrix for Logistic Regression')
y_pred = cross_val_predict(DecisionTreeClassifier(),X,Y,cv=10)
sns.heatmap(confusion_matrix(Y,y_pred),ax=ax[1,2],annot=True,fmt='2.0f')
ax[0,0].set_title('Perceptron')
y_pred = cross_val_predict(Perceptron(),X,Y,cv=10)
sns.heatmap(confusion_matrix(Y,y_pred),ax=ax[0,1],annot=True,fmt='2.0f')
ax[1,2].set_title('Matrix for Decision Tree')
y_pred = cross_val_predict(GaussianNB(),X,Y,cv=10)
sns.heatmap(confusion_matrix(Y,y_pred),ax=ax[2,0],annot=True,fmt='2.0f')
ax[2,0].set_title('Matrix for Naive Bayes')
plt.subplots_adjust(hspace=0.2,wspace=0.2)
plt.show()
```



In [ ]:
```
from sklearn.ensemble import VotingClassifier
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Perceptron
clf1 = LogisticRegression()
clf2 = SVC(probability=True)
clf3 = RandomForestClassifier()
clf4 = Perceptron()
clf5 = GaussianNB()
clf6 = KNeighborsClassifier(n_neighbors=5)
clf7 = DecisionTreeClassifier(random_state=0)
```

```
ensemble_clf = VotingClassifier(estimators=[
                                  ('lr', clf1),
                                  ('svc', clf2),
                                  ('rf', clf3),
                                  ('per', clf4),
                                  ('nb',clf5),
                                  ('knn',clf6),
                                  ('dt',clf7)
                                ],
                                voting='hard')
ensemble_clf.fit(train_X, train_Y)
print('The accuracy for ensembled model is:', ensemble_clf.score(test_X, test_Y))
cross = cross_val_score(ensemble_clf, X, Y, cv=10, scoring="accuracy")
print('The cross validated score is:', cross.mean())
```

```
The accuracy for ensembled model is: 0.9505016722408027
The cross validated score is: 0.9492068474297992
```

In [ ]:

In [ ]: