Project title : Brain-Stroke Prediction

Author : Venkat Jagadeesh Jampani

Class : DSC630 T302 2231-Fall2022

Course/section : Predictive Analytics

Date : Feb 12, 2023

Project Milestone-4

(Brain-Stroke Prediction)

Note: Milestone-4 content is added on top of milestone-3. All updates and milestone-4 is in blue color font.

Introduction:

The topic that I selected is related to the prediction of the Brain-Stroke. Recently I have seen a lot of people in my circle and friends experienced a stroke. A stroke, sometimes called a brain attack, occurs when something blocks blood supply to part of the brain or when a blood vessel in the brain bursts. In either case, parts of the brain become damaged or die. There are two main causes of stroke: a blocked artery (ischemic stroke) or leaking or bursting of a blood vessel (hemorrhagic stroke). Some people may have only a temporary disruption of blood flow to the brain, known as a transient ischemic attack (TIA), that doesn't cause lasting symptoms.

According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. A stroke can cause lasting brain damage, long-term disability, and/or even death. The brain controls our movements, stores our memories, and is the source of our thoughts, emotions, and language. The brain also controls many functions of the body, like breathing and digestion. To work properly, brain needs oxygen. Your arteries deliver oxygen-rich blood to all parts of your brain. If something happens to block the flow of blood, brain cells start to die within minutes**,** because they can't get oxygen. This causes a stroke.

Symptoms:

- Sudden numbness or weakness in the face, arm, or leg, especially on one side of the body.
- Sudden confusion, trouble speaking, or difficulty understanding speech.
- Sudden trouble seeing in one or both eyes.
- Sudden trouble walking, dizziness, loss of balance, or lack of coordination.
- Sudden severe headache with no known cause.

Facts & Stats per CDC:

- In 2020, 1 in 6 deaths from cardiovascular disease was due to stroke.
- Every 40 seconds, someone in the United States has a stroke. Every 3.5 minutes, someone dies of stroke.2
- Every year, more than 795,000 people in the United States have a stroke. About 610,000 of these are first or new strokes.
- About 185,000 strokes—nearly 1 in 4—are in people who have had a previous stroke.2
- About 87% of all strokes are ischemic strokes, in which blood flow to the brain is blocked.
- Stroke-related costs in the United States came to nearly $53 billion between 2017 and 2018. This total includes the cost of health care services, medicines to treat stroke and missed days of work.
- Stroke is a leading cause of 1. Serious long-term disability .2. Stroke reduces mobility in more than half of stroke survivors age 65 and older.
- Stroke is a leading cause of death for Americans, but the risk of having a stroke varies with race and ethnicity.
- Risk of having a first stroke is nearly twice as high for Blacks as for Whites, and Blacks have the highest rate of death due to stroke.
- Though stroke death rates have declined for decades among all race/ethnicities, Hispanics have seen an increase in death rates since 2013.

Some of the risk factors that can possibly increase one's risk of stroke include:

- Age
- Sex
- Race/Ethnicity
- High Blood Pressure
- High Cholesterol
- Heart Disease
- Diabetes
- Unhealthy Diet/Physical Inactivity
- Obesity
- Tobacco/Alcohol Use

Project Proposal/Problem Statement:

1. How can we predict if a person is likely to experience a brain-stroke?
2. What's the likeliness of a brain-stroke based on his/her available daily routine and medical history?

The project proposal therefore is "Brain-Stroke Prediction" using common risk factors and applying predictive modeling techniques.

Dataset Used:

The dataset that I am using for this project is collected from Kaggle.com. This dataset is one among the popular data from Kaggle with about 12 attributes and 5110 observations of health details. This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relevant information about the patient. Here is the link for the data source being used.

URL: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset?resource=download

Attribute details of the dataset:

- id: unique identifier
- gender: "Male", "Female" or "Other"
- age: age of the patient
- hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
- heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
- ever_married: "No" or "Yes"
- work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
- Residence_type: "Rural" or "Urban"
- avg_glucose_level: average glucose level in blood
- bmi: body mass index
- smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"*
- stroke: 1 if the patient had a stroke or 0 if not

***Note**: "Unknown" in smoking_status means that the information is unavailable for this patient.

Models to be used/Approach:

In this project, I will follow CRISP-DM model for data exploration/understanding, modeling, and evaluation. Will perform the following activities in this process, Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. Python programming language will be used for data load, exploration, fitting the models and testing the accuracy along with some of the machine learning libraries that might be required.

Firstly, EDA will be performed to better understand the data and compute descriptive statistics followed by visualizations representing the dataset.

Later, various machine learning algorithms will be employed to examine and choose the best possible prediction model. Some of them include Logistic Regression, K Nearest Neighbor (KNN), Decision Tree, Naïve Bayes, Support Vector Machine (SVM) and Random Forest Classifier. I would try to apply and check the accuracy of these models as these models are mostly industry standard and widely used for building predictive models.

Data versus Expectations:

Will I be able to answer the questions I want to answer with the data I have?

I feel confident that I shall be able to get the answer to the questions; I want with the dataset that I selected. Based on initial analysis at the dataset, 11 useful features (age, heart disease hist, gender, avg glucose level, BMI, stroke, smoking status, hypertension) are available (excluding id) which should be very much enough data, to build a reasonable accurate model in predicting the likeliness of brain stroke in the targeted population. I should be able to create few visualizations and correlations between variables and apply different models (like K Nearest Neighbor (KNN), Logistic Regression, Decision Tree, Support Vector Machine (SVM)) on top of this data to be able to come up with a best performing model and answer my research questions of stroke likeliness based on an individual medical history/available data.

However, it would be much better to have few more variables like alcohol consumption, physical activity, nutrition/diet, family history etc., with possibly more observations/rows which are currently missing in this dataset.

Motivation:

<u>What is the motivation for selecting the data set?</u>

According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. A stroke can cause lasting brain damage, long-term disability, and/or even death. The brain controls our movements, stores our memories, and is the source of our thoughts, emotions, and language. The brain also controls many functions of the body, like breathing and digestion. To work properly, brain needs oxygen. Your arteries deliver oxygen-rich blood to all parts of your brain. If something happens to block the flow of blood, brain cells start to die within minutes, because they can’t get oxygen. This causes a stroke. Furthermore, if the model proves robust enough, we could analyze which variables are the most predictive, or see if some combination of variables is predictive. That information could spark questions to inspire further research, so that it can help to predict the brain stroke and preventive measures can be followed.

Visualizations:

<u>What visualizations are especially useful for explaining my data?</u>

I plan to employ pie charts first on different categorical variables to see the distributions like gender (male/female), hypertension history (y/n), heart disease history (y/n), marriage (y/n), residence (urban/rural), smoking, stroke history (y/n) , avg_BMI, avg_glucose_level and so on. This will give me an idea on how well the data is balanced across these variables.

I would also use histograms/bar plots on different numerical variables like age, avg glucose level and bmi to see the variations/distributions for the same.

Heatmaps shall be used in finding correlations between categorical/numerical variables vs stroke will also help a great deal to explain my data much better.

<u>Do I need to adjust the data and/or the driving questions?</u>

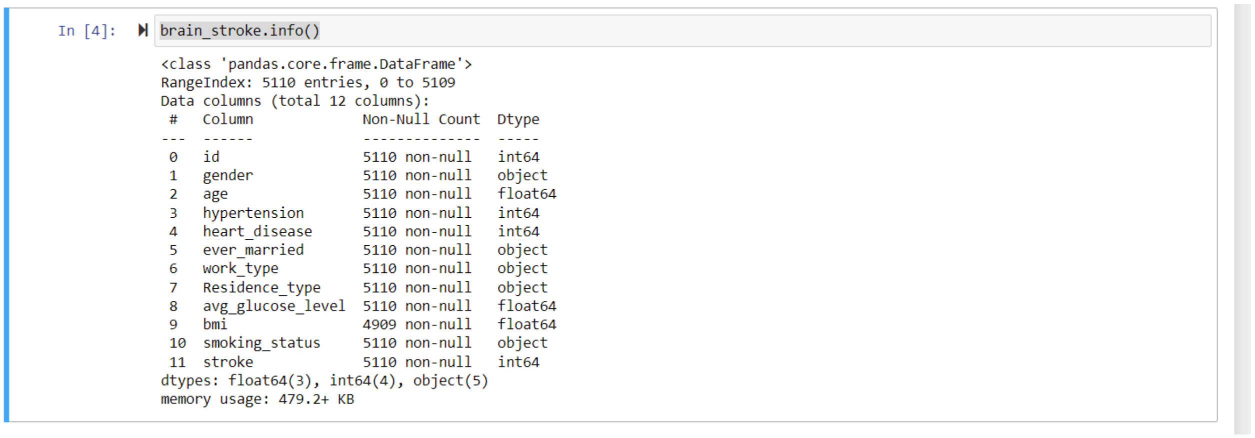
Based on exploratory data analysis (EDA), visualizations and initial model results, there is a possibility to imbalances in the train data set and might need some rebalancing of the classes. I plan to employ the machine learning technique SMOTE (Synthetic Minority Oversampling Technique) to overcome this issue if the training data set is imbalanced which usually happens if there are more observations of one categorical variable type and very few for other type.

Irrespective of this, I should be still using the required tools/techniques and still be able to address the questions with the given dataset.

<u>Explain your process for prepping the data</u>

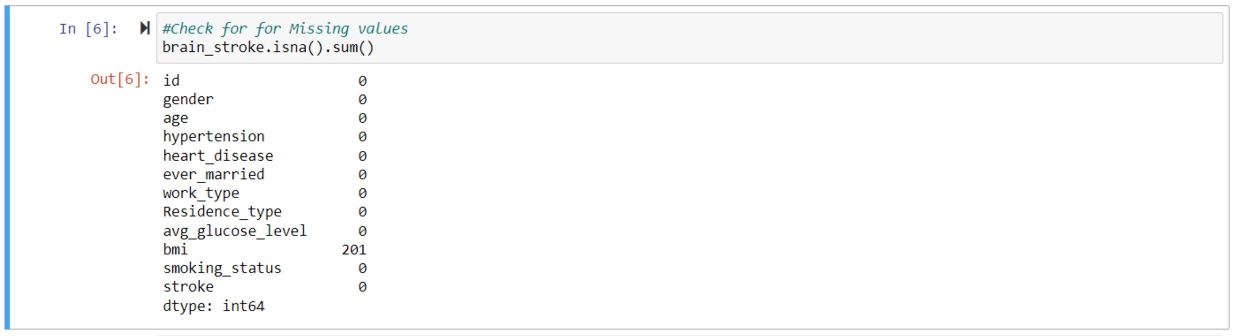
Import the Kaggle sourced dataset -> "healthcare-dataset-stroke-data.csv" using pandas read csv utility.

a. Check basic info to verify attributes, no. of observations and data types.



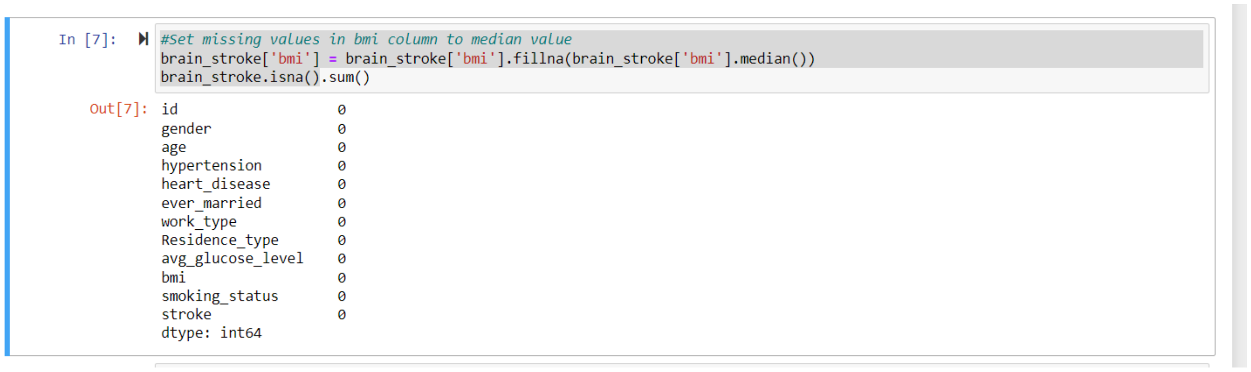
```
In [4]:  brain_stroke.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

b. Check for missing values in the dataset.



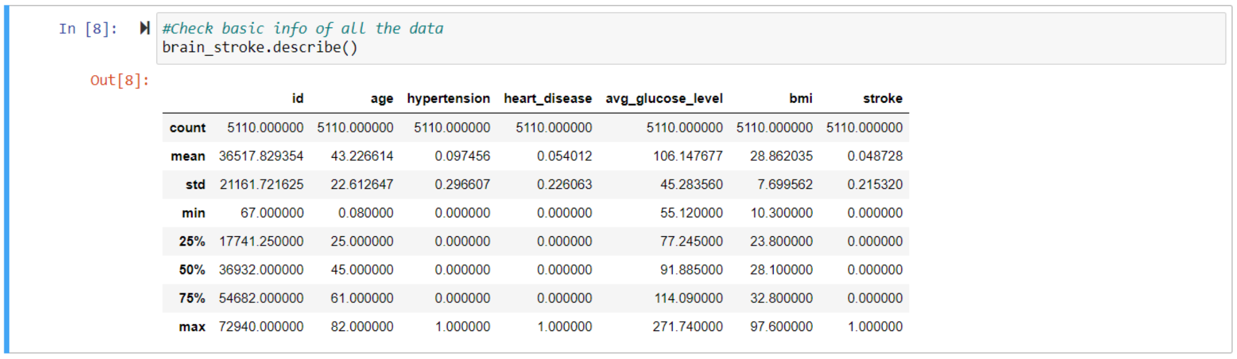
```
In [6]:  #Check for for Missing values
         brain_stroke.isna().sum()

Out[6]:  id                     0
         gender                 0
         age                    0
         hypertension           0
         heart_disease          0
         ever_married           0
         work_type              0
         Residence_type         0
         avg_glucose_level      0
         bmi                  201
         smoking_status         0
         stroke                 0
         dtype: int64
```

c. Only “bmi” attribute has 201 missing values. So, will set the missing bmi rows to median values.



```
In [7]:  #Set missing values in bmi column to median value
         brain_stroke['bmi'] = brain_stroke['bmi'].fillna(brain_stroke['bmi'].median())
         brain_stroke.isna().sum()

Out[7]:  id                   0
         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
```

d. Check basic info of the dataset.

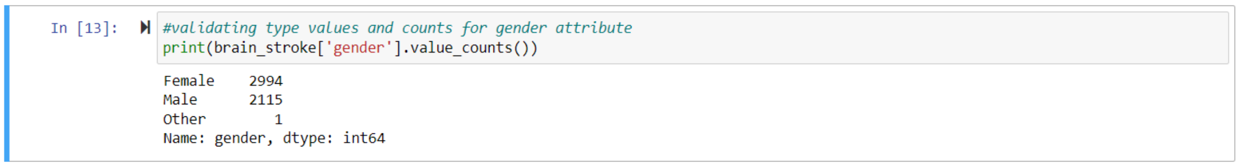


```
In [8]:  #Check basic info of all the data
         brain_stroke.describe()

Out[8]:
```

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.862035	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.699562	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.800000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	32.800000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

Observation : 4.87% of the population in this dataset had stroke.

1. On the data set : Histograms and Pie Charts are plotted for numerical and categorical attributes like "age", "avg_glucose_level", "bmi", martial status respectively
2. Validate type values and counts for gender attribute and a review revealed an outlier here which is removed.

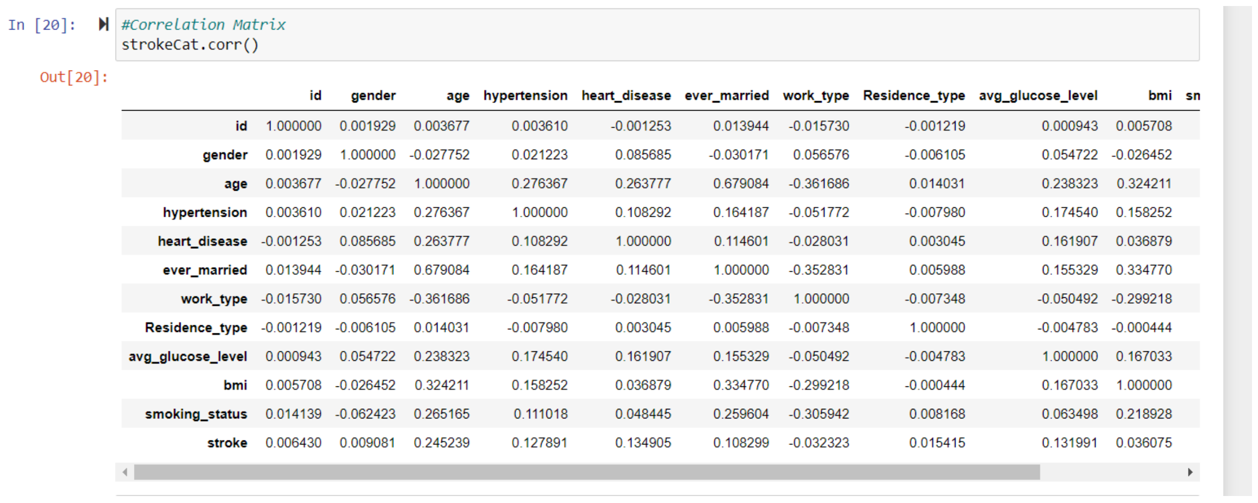


```
In [13]:  #validating type values and counts for gender attribute
          print(brain_stroke['gender'].value_counts())

          Female    2994
          Male      2115
          Other        1
          Name: gender, dtype: int64
```

e. Create a list of columns that have categorical data.

f. Create dummy variables for all categorical columns.

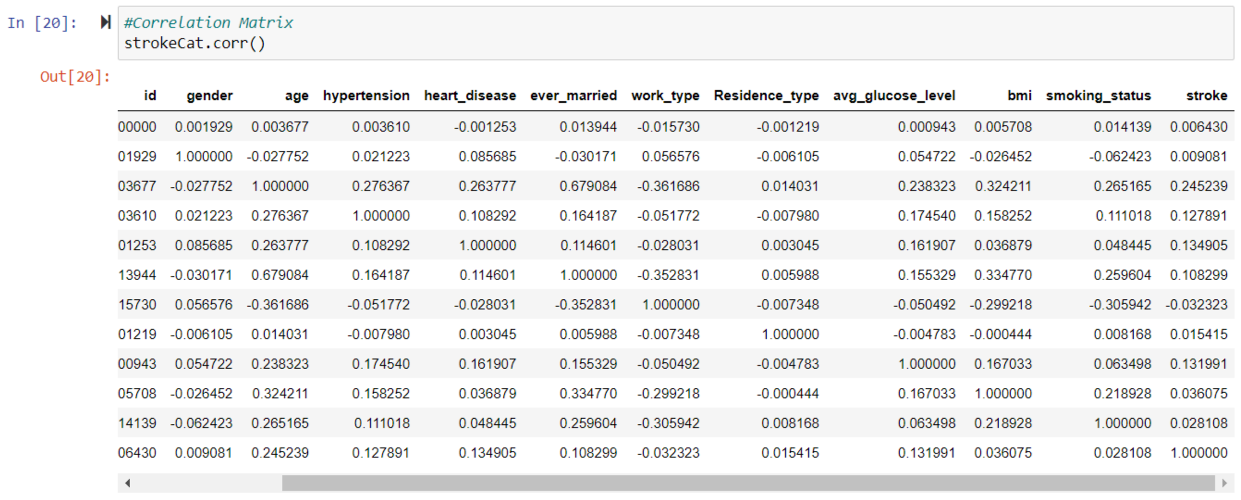
g. Validate categorical value conversion and create a correlation matrix.



```
In [20]:  #Correlation Matrix
          strokeCat.corr()
```

Out[20]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	sn
id	1.000000	0.001929	0.003677	0.003610	-0.001253	0.013944	-0.015730	-0.001219	0.000943	0.005708	
gender	0.001929	1.000000	-0.027752	0.021223	0.085685	-0.030171	0.056576	-0.006105	0.054722	-0.026452	
age	0.003677	-0.027752	1.000000	0.276367	0.263777	0.679084	-0.361686	0.014031	0.238323	0.324211	
hypertension	0.003610	0.021223	0.276367	1.000000	0.108292	0.164187	-0.051772	-0.007980	0.174540	0.158252	
heart_disease	-0.001253	0.085685	0.263777	0.108292	1.000000	0.114601	-0.028031	0.003045	0.161907	0.036879	
ever_married	0.013944	-0.030171	0.679084	0.164187	0.114601	1.000000	-0.352831	0.005988	0.155329	0.334770	
work_type	-0.015730	0.056576	-0.361686	-0.051772	-0.028031	-0.352831	1.000000	-0.007348	-0.050492	-0.299218	
Residence_type	-0.001219	-0.006105	0.014031	-0.007980	0.003045	0.005988	-0.007348	1.000000	-0.004783	-0.000444	
avg_glucose_level	0.000943	0.054722	0.238323	0.174540	0.161907	0.155329	-0.050492	-0.004783	1.000000	0.167033	
bmi	0.005708	-0.026452	0.324211	0.158252	0.036879	0.334770	-0.299218	-0.000444	0.167033	1.000000	
smoking_status	0.014139	-0.062423	0.265165	0.111018	0.048445	0.259604	-0.305942	0.008168	0.063498	0.218928	
stroke	0.006430	0.009081	0.245239	0.127891	0.134905	0.108299	-0.032323	0.015415	0.131991	0.036075	



```
In [20]:  #Correlation Matrix
          strokeCat.corr()
```

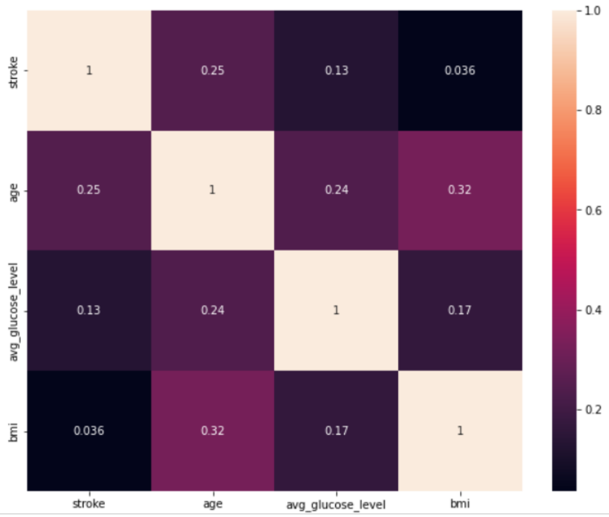
Out[20]:

id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
00000	0.001929	0.003677	0.003610	-0.001253	0.013944	-0.015730	-0.001219	0.000943	0.005708	0.014139	0.006430
01929	1.000000	-0.027752	0.021223	0.085685	-0.030171	0.056576	-0.006105	0.054722	-0.026452	-0.062423	0.009081
03677	-0.027752	1.000000	0.276367	0.263777	0.679084	-0.361686	0.014031	0.238323	0.324211	0.265165	0.245239
03610	0.021223	0.276367	1.000000	0.108292	0.164187	-0.051772	-0.007980	0.174540	0.158252	0.111018	0.127891
01253	0.085685	0.263777	0.108292	1.000000	0.114601	-0.028031	0.003045	0.161907	0.036879	0.048445	0.134905
13944	-0.030171	0.679084	0.164187	0.114601	1.000000	-0.352831	0.005988	0.155329	0.334770	0.259604	0.108299
15730	0.056576	-0.361686	-0.051772	-0.028031	-0.352831	1.000000	-0.007348	-0.050492	-0.299218	-0.305942	-0.032323
01219	-0.006105	0.014031	-0.007980	0.003045	0.005988	-0.007348	1.000000	-0.004783	-0.000444	0.008168	0.015415
00943	0.054722	0.238323	0.174540	0.161907	0.155329	-0.050492	-0.004783	1.000000	0.167033	0.063498	0.131991
05708	-0.026452	0.324211	0.158252	0.036879	0.334770	-0.299218	-0.000444	0.167033	1.000000	0.218928	0.036075
14139	-0.062423	0.265165	0.111018	0.048445	0.259604	-0.305942	0.008168	0.063498	0.218928	1.000000	0.028108
06430	0.009081	0.245239	0.127891	0.134905	0.108299	-0.032323	0.015415	0.131991	0.036075	0.028108	1.000000

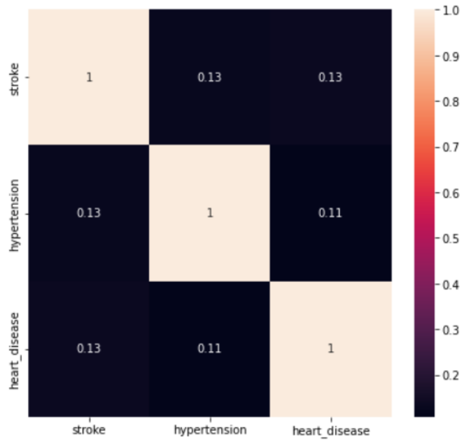
h. Plot Correlation heatmap against numerical and categorical attributes.

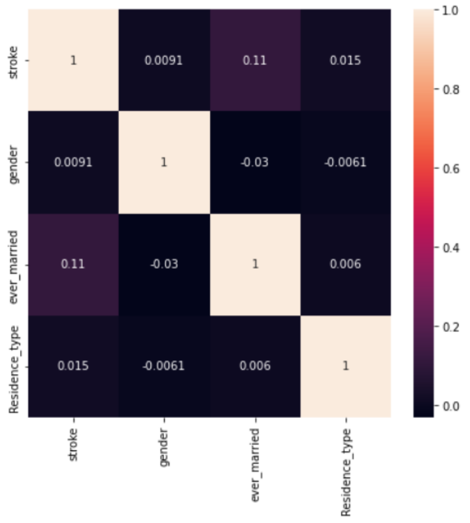
In [22]:
```python
#plotting Correlation heat map against numeric attributes : 'stroke', 'age', 'avg_glucose_level', 'bmi'
fig, ax = plt.subplots(figsize=(10,8))
sns.heatmap(strokeCat[['stroke', 'age', 'avg_glucose_level', 'bmi']].corr(),annot=True)
```

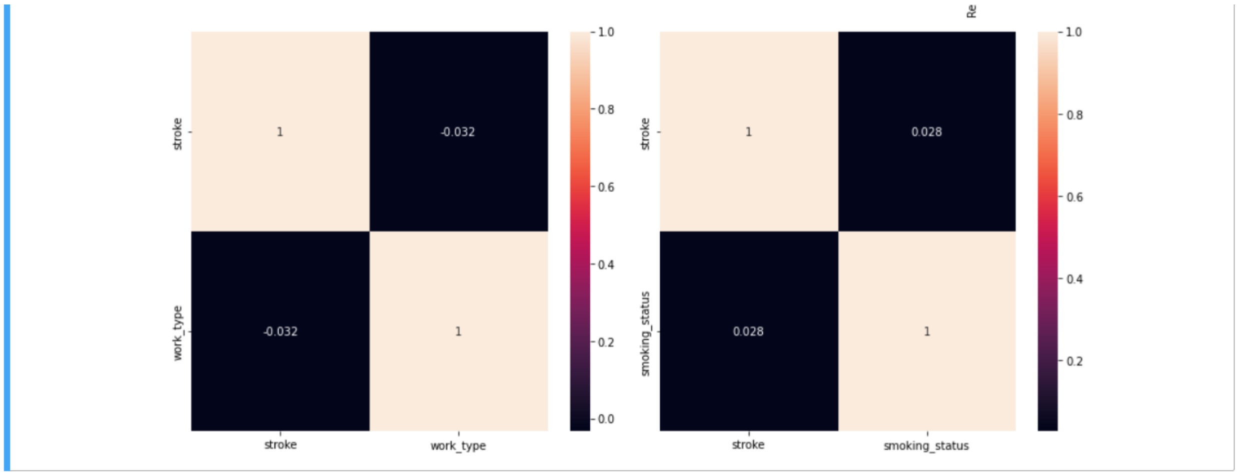
Out[22]:
```
<AxesSubplot:>
```



In [23]:
```python
#Correlation heat map against categorical attributes
# 1. ['stroke', 'hypertension', 'heart_disease']
# 2. ['stroke', 'gender', 'ever_married', 'Residence_type']
# 3. ['stroke', 'work_type']
# 4. ['stroke', 'smoking_status']
fig, ax = plt.subplots(2,2, figsize = (12,12))
((ax1, ax2), (ax3, ax4)) = ax

# the "no_" attributes is the opposite to the "yes_" attributes so the correlation to stroke will be the same but negative.
sns.heatmap(strokeCat[['stroke', 'hypertension', 'heart_disease']].corr(),annot=True, ax=ax1)
sns.heatmap(strokeCat[['stroke', 'gender', 'ever_married', 'Residence_type']].corr(),annot=True, ax=ax2)
sns.heatmap(strokeCat[['stroke', 'work_type']].corr(),annot=True, ax=ax3)
sns.heatmap(strokeCat[['stroke', 'smoking_status']].corr(),annot=True, ax=ax4)

plt.tight_layout()
plt.show()
```







i. Check various other plots against other key attributes for any observations.(covered in the final code)
j. Prepare the dataset for modeling by splitting the dataset into train and test datasets (75/25) and **apply knn modeling** with standard scalar and hyper parameters for grid search.(next steps)

<u>Evaluation of results:</u>

Based on the models used and carefully examining them, I expect to curate an effective and most accurate and sensible working model (with train and test datasets) to predict the likeliness of stroke in patients based on the available medical history of patients from the given dataset (sourced). I would also check and review the train data set before fitting the models to ensure the classes are not imbalanced anywhere in the train data set which incase might need to rebalance them before fitting the actual models for evaluation.

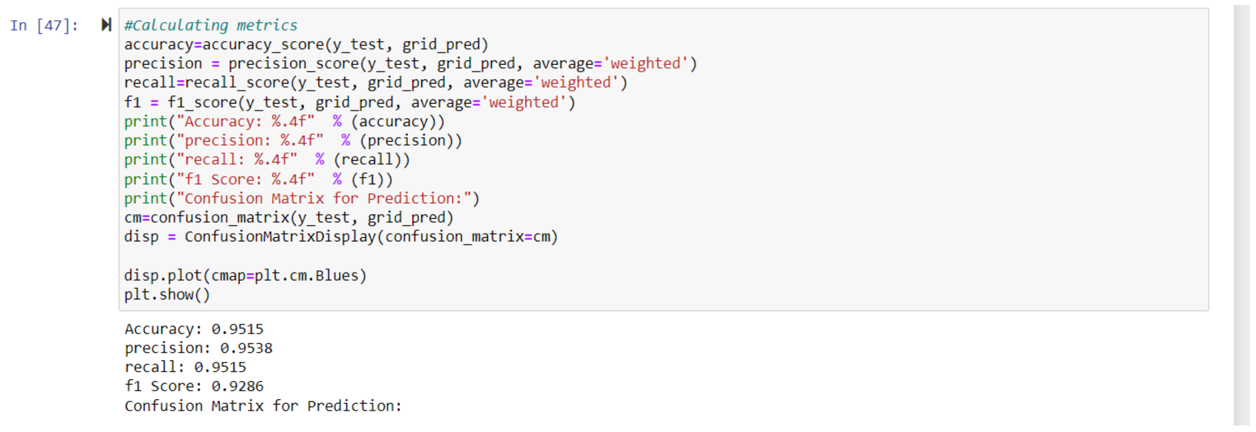
Model versus expectations:

<u>Do I need to adjust my model/evaluation choices?</u>

For the dataset I have selected, I feel that I don't really need to adjust or change to any other new models apart from the ones mentioned above. The models mentioned are most likely enough to come up with a prediction model for this kind of dataset. The review of the train dataset feed to the model should help in gauging. Applying/fitting the models mentioned earlier and evaluating their accuracy scores/performance against the train data set should most likely yield a better model with best accuracy results for the dataset being used here. Having said that, I am open to add any other model based on the analysis and outcome of the results in the next phases, even though I feel that is highly unlikely.

Build and evaluate at least one model

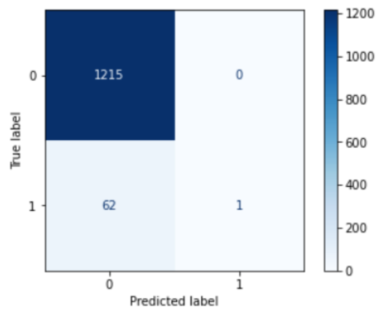
I have applied KNN modeling with original train and test data using standard scalar and creating grid search with fitting the train and test data. After that, check the accuracy score and calculate the metrics.



```
In [47]:
#Calculating metrics
accuracy=accuracy_score(y_test, grid_pred)
precision = precision_score(y_test, grid_pred, average='weighted')
recall=recall_score(y_test, grid_pred, average='weighted')
f1 = f1_score(y_test, grid_pred, average='weighted')
print("Accuracy: %.4f"  % (accuracy))
print("precision: %.4f"  % (precision))
print("recall: %.4f"  % (recall))
print("f1 Score: %.4f"  % (f1))
print("Confusion Matrix for Prediction:")
cm=confusion_matrix(y_test, grid_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)

disp.plot(cmap=plt.cm.Blues)
plt.show()
```

```
Accuracy: 0.9515
precision: 0.9538
recall: 0.9515
f1 Score: 0.9286
Confusion Matrix for Prediction:
```

```
Accuracy: 0.9515
precision: 0.9538
recall: 0.9515
f1 Score: 0.9286
Confusion Matrix for Prediction:
```



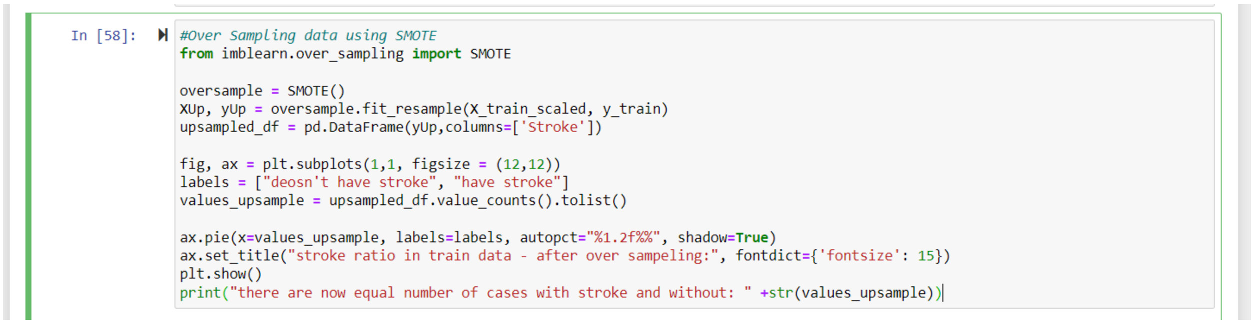
Observation: I see that knn model has resulted in very high accuracy/precision/recall and f1 scores - all of which are greater than 95% except that f1 score is at 92%.

This high accuracy could be a result of imbalanced dataset (95% negative outcomes and 5% positive outcomes of stroke).

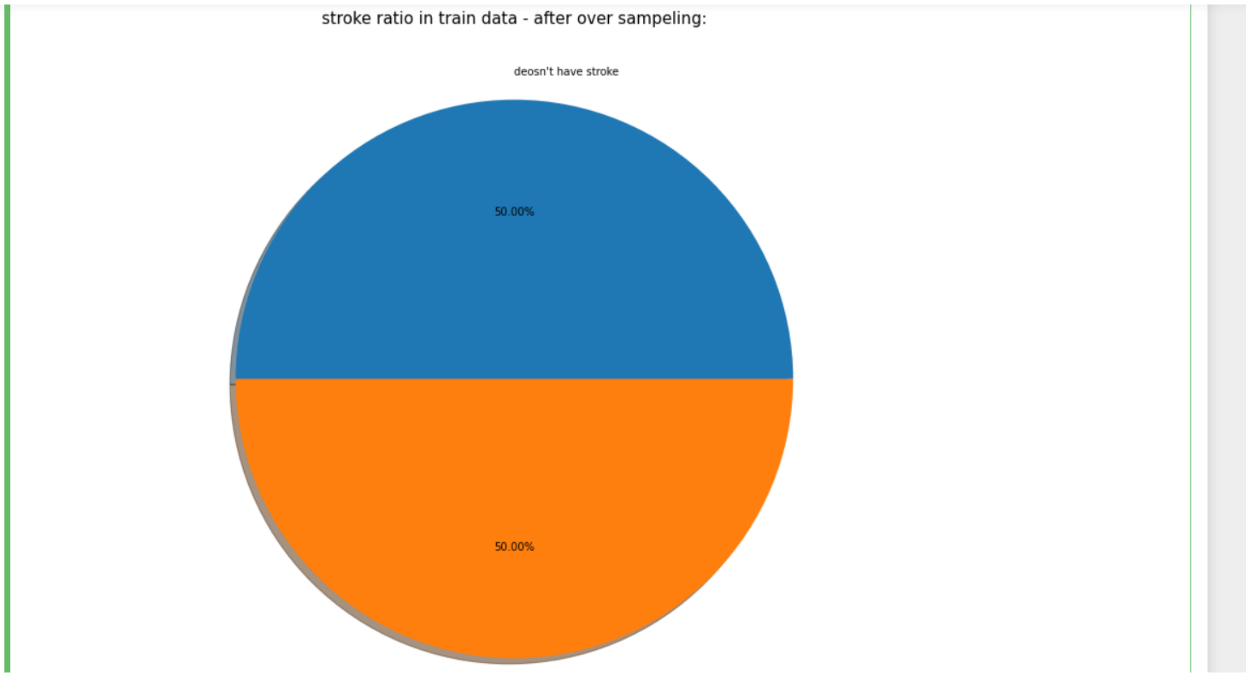
Interpret your results

Based on the initial results, the high accuracy could very well be a result of imbalanced datasets being used (having 95% negative outcomes and 5% positive outcomes for brain stroke).

As a result, I am going to use SMOTE to oversample my unbalanced positive outcome.

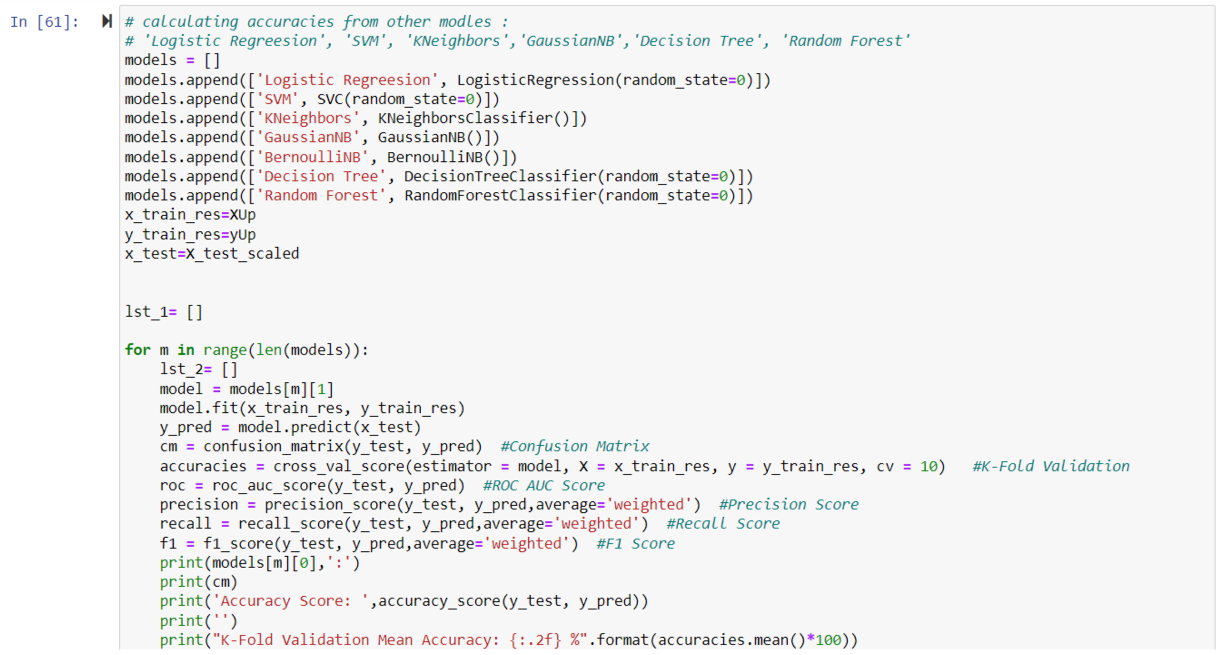


```
In [58]:
#Over Sampling data using SMOTE
from imblearn.over_sampling import SMOTE

oversample = SMOTE()
XUp, yUp = oversample.fit_resample(X_train_scaled, y_train)
upsampled_df = pd.DataFrame(yUp,columns=['Stroke'])

fig, ax = plt.subplots(1,1, figsize = (12,12))
labels = ["deosn't have stroke", "have stroke"]
values_upsample = upsampled_df.value_counts().tolist()

ax.pie(x=values_upsample, labels=labels, autopct="%1.2f%%", shadow=True)
ax.set_title("stroke ratio in train data - after over sampeling:", fontdict={'fontsize': 15})
plt.show()
print("there are now equal number of cases with stroke and without: " +str(values_upsample))
```

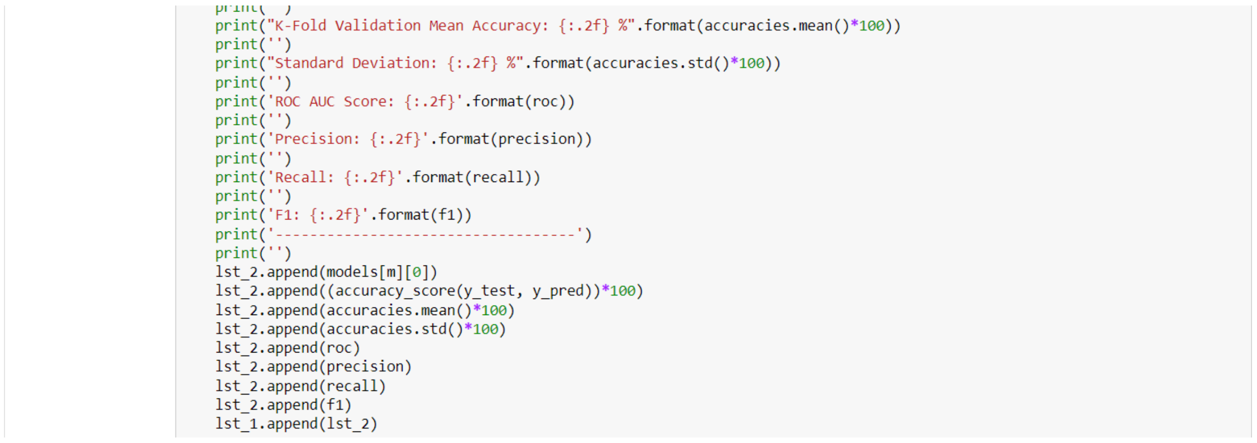


```
there are now equal number of cases with stroke and without: [3645, 3645]
```

Fit few other models to see individual accuracies.



```
In [61]:
# calculating accuracies from other modles :
# 'Logistic Regreesion', 'SVM', 'KNeighbors','GaussianNB','Decision Tree', 'Random Forest'
models = []
models.append(['Logistic Regreesion', LogisticRegression(random_state=0)])
models.append(['SVM', SVC(random_state=0)])
models.append(['KNeighbors', KNeighborsClassifier()])
models.append(['GaussianNB', GaussianNB()])
models.append(['BernoulliNB', BernoulliNB()])
models.append(['Decision Tree', DecisionTreeClassifier(random_state=0)])
models.append(['Random Forest', RandomForestClassifier(random_state=0)])
x_train_res=XUp
y_train_res=yUp
x_test=X_test_scaled


lst_1= []

for m in range(len(models)):
    lst_2= []
    model = models[m][1]
    model.fit(x_train_res, y_train_res)
    y_pred = model.predict(x_test)
    cm = confusion_matrix(y_test, y_pred)  #Confusion Matrix
    accuracies = cross_val_score(estimator = model, X = x_train_res, y = y_train_res, cv = 10)   #K-Fold Validation
    roc = roc_auc_score(y_test, y_pred)  #ROC AUC Score
    precision = precision_score(y_test, y_pred,average='weighted')  #Precision Score
    recall = recall_score(y_test, y_pred,average='weighted')  #Recall Score
    f1 = f1_score(y_test, y_pred,average='weighted')  #F1 Score
    print(models[m][0],':')
    print(cm)
    print('Accuracy Score: ',accuracy_score(y_test, y_pred))
    print('')
    print("K-Fold Validation Mean Accuracy: {:.2f} %".format(accuracies.mean()*100))
```



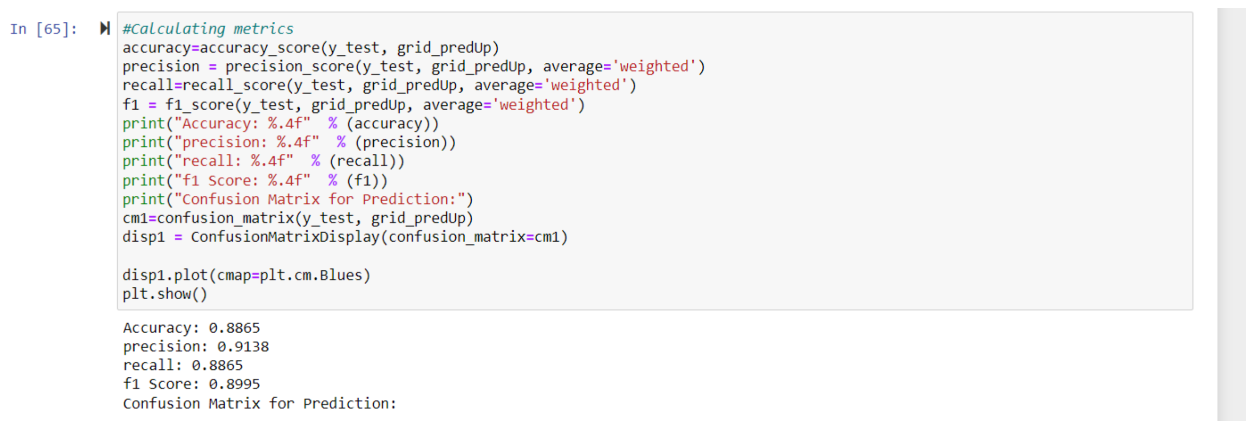
```
    print("K-Fold Validation Mean Accuracy: {:.2f} %".format(accuracies.mean()*100))
    print('')
    print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
    print('')
    print('ROC AUC Score: {:.2f}'.format(roc))
    print('')
    print('Precision: {:.2f}'.format(precision))
    print('')
    print('Recall: {:.2f}'.format(recall))
    print('')
    print('F1: {:.2f}'.format(f1))
    print('-----------------------------------')
    print('')
    lst_2.append(models[m][0])
    lst_2.append((accuracy_score(y_test, y_pred))*100)
    lst_2.append(accuracies.mean()*100)
    lst_2.append(accuracies.std()*100)
    lst_2.append(roc)
    lst_2.append(precision)
    lst_2.append(recall)
    lst_2.append(f1)
    lst_1.append(lst_2)
```

```
Logistic Regreesion :
[[899 316]
 [ 15  48]]
Accuracy Score:  0.741001564945227



K-Fold Validation Mean Accuracy: 78.18 %
Standard Deviation: 1.74 %
ROC AUC Score: 0.75
Precision: 0.94
Recall: 0.74
```

```
F1: 0.81
------------------------------------
SVM :
[[943 272]
 [ 26  37]]
Accuracy Score:  0.7668231611893583
K-Fold Validation Mean Accuracy: 84.29 %
Standard Deviation: 1.41 %
ROC AUC Score: 0.68
Precision: 0.93
Recall: 0.77
F1: 0.83
------------------------------------
KNeighbors :
[[1027  188]
 [  44   19]]
Accuracy Score:  0.8184663536776213
K-Fold Validation Mean Accuracy: 90.99 %
Standard Deviation: 1.02 %
ROC AUC Score: 0.57
Precision: 0.92
Recall: 0.82
F1: 0.86
------------------------------------
GaussianNB :
[[873 342]
 [ 16  47]]
Accuracy Score:  0.7198748043818466
K-Fold Validation Mean Accuracy: 77.06 %
Standard Deviation: 1.39 %
ROC AUC Score: 0.73
Precision: 0.94
Recall: 0.72
F1: 0.80
------------------------------------
BernoulliNB :
[[684 531]
 [  9  54]]
Accuracy Score:  0.5774647887323944
K-Fold Validation Mean Accuracy: 72.07 %
Standard Deviation: 1.04 %
ROC AUC Score: 0.71
Precision: 0.94
Recall: 0.58
F1: 0.69
------------------------------------
Decision Tree :
[[1124   91]
 [  46   17]]
Accuracy Score:  0.8928012519561815
K-Fold Validation Mean Accuracy: 91.60 %
Standard Deviation: 3.27 %
ROC AUC Score: 0.60
Precision: 0.92
```

```
Recall: 0.89
F1: 0.91
-----------------------------------
Random Forest :
[[1177   38]
 [  56    7]]
Accuracy Score:  0.9264475743348983
K-Fold Validation Mean Accuracy: 96.49 %
Standard Deviation: 2.51 %
ROC AUC Score: 0.54
Precision: 0.92
Recall: 0.93
F1: 0.92
-----------------------------------
```

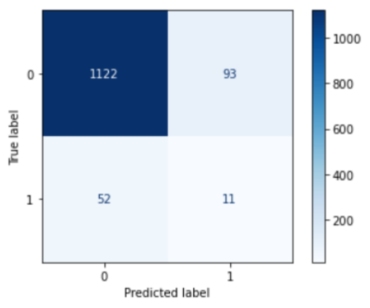
Rerun the earlier KNN grid search again with train data to balance classes using SMOTE and calculate the metrics to derive conclusions.



In [65]:

```python
#Calculating metrics
accuracy=accuracy_score(y_test, grid_predUp)
precision = precision_score(y_test, grid_predUp, average='weighted')
recall=recall_score(y_test, grid_predUp, average='weighted')
f1 = f1_score(y_test, grid_predUp, average='weighted')
print("Accuracy: %.4f"  % (accuracy))
print("precision: %.4f"  % (precision))
print("recall: %.4f"  % (recall))
print("f1 Score: %.4f"  % (f1))
print("Confusion Matrix for Prediction:")
cm1=confusion_matrix(y_test, grid_predUp)
disp1 = ConfusionMatrixDisplay(confusion_matrix=cm1)

disp1.plot(cmap=plt.cm.Blues)
plt.show()
```

```
Accuracy: 0.8865
precision: 0.9138
recall: 0.8865
f1 Score: 0.8995
Confusion Matrix for Prediction:
```



Risks:

Following the potential risks involved in this project:

- This project assumes that the dataset sourced is valid and accurate.
- There is always a chance of data loss due to incompleteness or extremities/anomalies.
- The number of observations in the dataset used is low (5110) for this study.
- This dataset does not include all the risk factors of a stroke like diabetes, alcohol consumption, physical activity etc., hence the model cannot be fully accurate.
- The data set is updated last in the year 2018. So the perdition based on the old data might not predict the current and future results.

Contingency Plan:

If for some reason, the project cannot be completed with the available data or shows no sign of predictive potential, then I would take a portion of data or consider a new similar dataset altogether to come up with a possible predictive outcome/model.

If CRISP-DM model, is not fitting the selected data, I will consider SEMMA, KDD.

KDD : Knowledge Discovery in Databases or KDD, for short, is a method of how specialists can extract patterns and/or required information from data. It consists of five stages — Selection, Preprocessing, Transformation, Data Mining, and Interpretation/Evaluation.

SEMMA : SEMMA has a similar structure to KDD, but as it does not focus as heavily on data-specific stages, it is easier to apply to general Data Science tasks. Also, it has strictly cyclic nature, unlike KDD. SEMMA is an acronym that stands for Sample, Explore, Modify, Model and Access.

In the backup plan, the following data source can be considered, which are in similar area of prediction. :

Similar Datasets: Topics

- [**HIGHLIGHTED**] CERN Electron Collision Data.
- Hepatitis C Dataset.
- Body Fat Prediction Dataset.
- Cirrhosis Prediction Dataset.
- Heart Failure Prediction Dataset.
- Wind Speed Prediction Dataset.

The details are available in the below link :

https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset?resource=download

Learning:

I anticipate and hope to learn the application of predictive modeling techniques on a given dataset and come up with the best possible accurate and effective model/outcome. Learn to come up with all the relevant information to be able to present to the management/teams to be able to understand and make informed decisions. Also, to learn and improve the approach and model deployment based on the inputs/review from peers.

Review:

- For Milestone-2 and later, I would like to perform peer reviews for following projects/peers.
    1. Naveen Bagam
    2. Eric Vance

Are my original expectations still reasonable?

I feel confident that the original expectations are still reasonable to be able to build a predictive model for brain stroke likeliness based on the variables/features available in the given dataset and the model approach I mentioned earlier. While the dataset falls short on the number of features and observations, it is still good enough to address the research questions and build a decent accurate model.

Begin to formulate a conclusion/recommendations

Based on the above model evaluations and interpreting the results, it is safe to formulate certain conclusion/recommendations:

1. Initial model (KNN) evaluation shows that it's not really identifying any positive outcome is it has 95% negative outcomes and 5% positive outcomes.
2. So the I planned to use SMOTE to balance my training set to include more values for positive outcome.
3. Several of the individual models like Logistics regression to Random Forest classifier had lesser precision (lesser by few percentage points) than original KNN Model which makes the case of KNN model application to be strong for this stroke prediction.
4. Retaining of the dataset with KNN using SMOTE should most likely yield better accuracy than other models and positive outcomes of stroke and therefore feel this is a better model.

References:

1. Centers for Disease Control and Prevention. Underlying Cause of Death, 1999–2018. CDC WONDER Online Database. Centers for Disease Control and Prevention; 2018. Accessed March 12, 2020.
2. Tsao CW, Aday AW, Almarzooq ZI, Alonso A, Beaton AZ, Bittencourt MS, et al. Heart Disease and Stroke Statistics—2022 Update: A Report From the American Heart Associationexternal icon. *Circulation*. 2022;145(8):e153–e639.
3. Jackson G, Chari K. National Hospital Care Survey Demonstration Projects: Stroke Inpatient Hospitalizationsexternal icon. *Natl Health Stat Report*. 2019 Nov;(132):1-11.
4. Fang J, Keenan NL, Ayala C, Dai S, Merritt R, Denny CH. Awareness of stroke warning symptoms—13 states and the District of Columbia, 2005. *MMWR*. 2008;57(18):481–5.
5. Kaggle, https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset?resource=download
6. KDD, SEMMA model details : https://medium.datadriveninvestor.com/data-science-project-management-methodologies-f6913c6b29eb