STATS/CSE 780 - Project Report

Xiaoran Xie (Student number: 400549859)

2023-12-11

Abstract

Stroke, as an epidemic that kills millions of people annually, representing a significant challenge in healthcare, has become a health issue that we need to be concerned about. Predicting stroke risk is important since early detection and intervention are key to reducing its prevalence. If the risk of stroke can be predicted based on an individual's health information, it will potentially save lives and reduce the impact on non-patients, as well as help patients to receive timely treatment and reduce mortality(Uma Venugopal 2023). Acccording to data preprocessing and data transformation, we cleaned the data by handling missing value and outlier, converted catagorical data to numerical data, and so on. By exploratory data analysis, we found the correlated factors between various predictors and the occurrence of stroke. We identified key factors influencing stroke risk and dived into the analysis by looking for the relationship. For stroke diagnosis prediction and classification, we trained two different machine learning models to do analysis and comparison. Mutilayer Perceptron has better accurancy, but whether it is a good matched model for our project to predict the stroke is still under consideration.

Introduction

This dataset(FEDESORIANO 2023) contains health information on 5110 patients with confirmed or undiagnosed stroke obtained from the Kaggle platform, which is detailed and helps to find potential risk factors. The main goal of the project is to find the factors most associated with stroke risk and to build a prediction model based on this information. By developing a predictive model based on these factors, we can provide doctors with a powerful tool for assessing stroke risk more accurately. This will help doctors more accurately assess a patient's stroke risk and give timely medical advice. In this project, we will use two supervised learning methods in machine learning to build risk prediction models that enable classification of stroke diagnoses.

Methods

In developing our stroke risk prediction model, we chose a combination of Random Forest and Multilayer Perceptron for classification. Random forest is an ensemble model composed of multiple decision trees. It can provide the features that have the greatest impact on the prediction results and can effectively handle non-linear relationships, which is very useful for understanding the data.

Although it is different from the high transparency and interpretability of decision trees, compared with multilayer perceptrons, it can still be tracked and understood through the information of tree nodes. Deep learning models, especially multi-layer perceptrons, are good at learning and representing complex non-linear relationships through the stacking of multiple hidden layers. And deep learning can perform well on large-scale data sets. However, there are many hyperparameters, such as the number of layers, number of neurons, learning rate, etc. Therefore, parameter adjustment and training are costly and require a large amount of computing resources, which will indirectly affect the progress of the experiment. Furthermore, the model often has lower interpretability compared to other traditional machine learning methods.

During Random Forest parameter tuning, we explored different splitting criteria (functions for measuring the quality of data splits), max_depth, and n_estimators (the number of trees in the forest). The choice of gini, entropy, and log_loss as splitting criteria was to assess the impact of different data impurity measurements on model performance. The choice of max_depth aimed to find the optimal depth that prevents overfitting while allowing the model to learn sufficient data. The choice of n_estimators was to determine the required number of trees while maintaining computational efficiency. Ultimately, the optimal parameter combination found was criterion: 'gini', max_depth: None, and n_estimators: 100, indicating that not limiting the depth of the decision trees and using 100 trees could achieve the best cross-validation score.

During Multilayer Perceptron parameter tuning, we considered different hidden_layer_sizes, activations (activation functions), solvers (optimization algorithms), learning_rate, and max_iter (maximum number of iterations). The choice of activation functions, such as identity, logistic, tanh, and relu define how neurons process input signals. The choice of lbfgs, sgd, and adam as solvers was to compare the efficiency and convergence of different optimization algorithms. The optimal parameters were activation: 'tanh', hidden_layer_sizes: 32, and learning_rate: 'invscaling', indicating that on the given dataset, using the hyperbolic tangent activation function, a single hidden layer with 32 neurons, and a learning rate that decreases with the number of iterations, could achieve the optimal cross-validation score.

Results

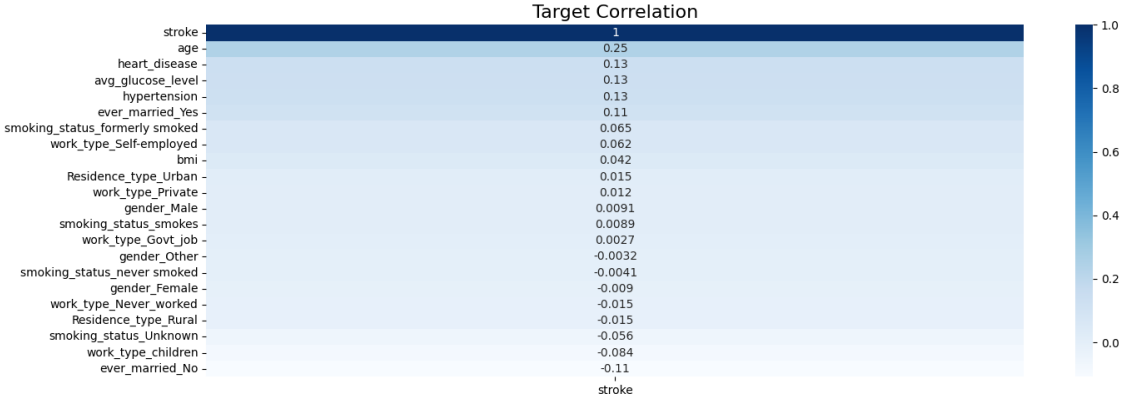
For handling missing BMI values in the dataset, we constructed a prediction model that exploited the correlation between the age and gender variables and BMI to estimate missing values from the relevant variables. Specifically, we implemented a pipeline(SHARMA 2023) containing two main processing steps: standardization of the data, followed by the application of a decision tree regression model. With this approach, we ensure consistency of model inputs. The standardization process ensures that each feature has the same weighting during model training, while the decision tree regressor provides us with a nonlinear way to predict missing values. We first excluded subsets containing missing BMI records. Then, the remaining complete data was utilized to train the pipeline model. We used the model to make predictions about missing BMI values and filled these predictions back into the original dataset at the appropriate locations. This method not only enhances the completeness of the data, but also improves the reliability of the subsequent analysis steps. We confirmed the validity of the method with updated missing value statistics.

Then we compared two encoding ways to transform categorical variables. The first method we used is one-hot encoding, a technique that creates binary columns for each category of a variable, which indicate the presence of a feature with a value of 1, while absence is marked with a value of 0. One-hot encoding does not impose any arbitrary ordering on the categories (e.g., 'gender-Male' is not greater than 'gender-Female'). This approach is beneficial when the categorical variable does not have an intrinsic order or ranking, and it ensures that the model does not attribute importance to the categories based on their encoded numerical values. The second method is label encoding, like enumeration, which assigns a unique integer to each category. Eventhough this method is more efficient and simpler, it assumes an ordinal relationship between the categories. So it could be misleading if applied to data where no such ordinal relationship exists. Combined with our data set, one-hot encoding better matches, and it proves to be more effective for our analysis than label encoding.

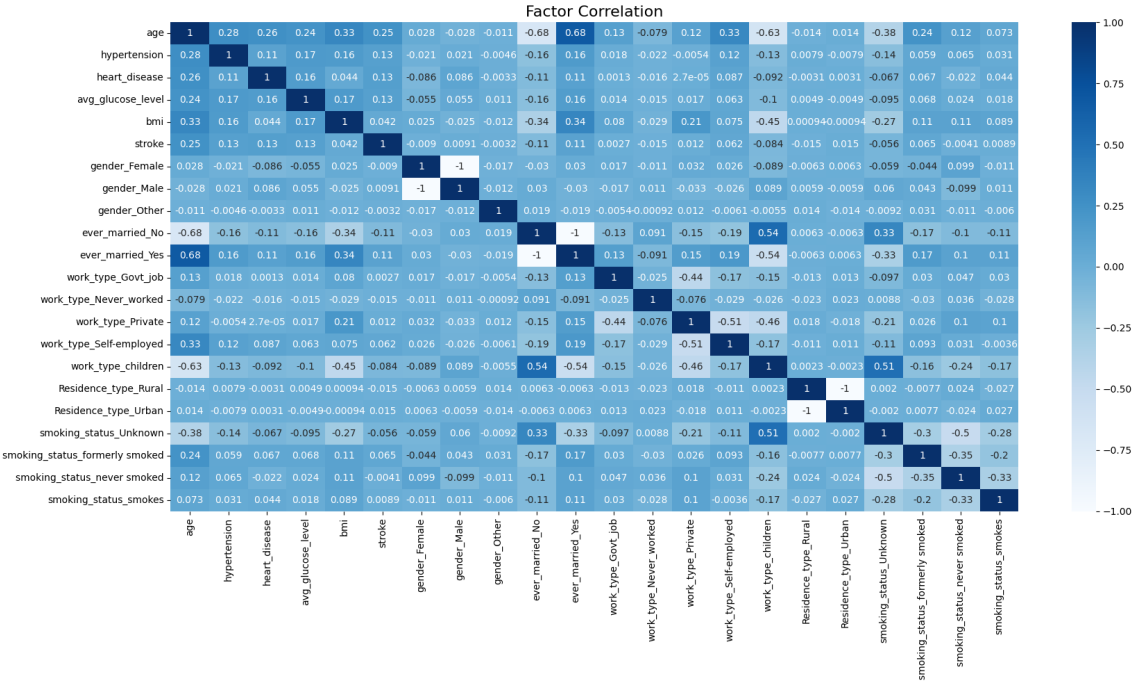
After we finished data type transformation, the original dataset was updated and suitbale to do exploratory data analysis.

The below bar chart visualizes the correlation coefficients between various predictors and the occurrence of stroke. Age shows the strongest positive correlation, indicating a higher stroke risk with increasing age. Heart disease and higher average glucose levels also show a moderate positive

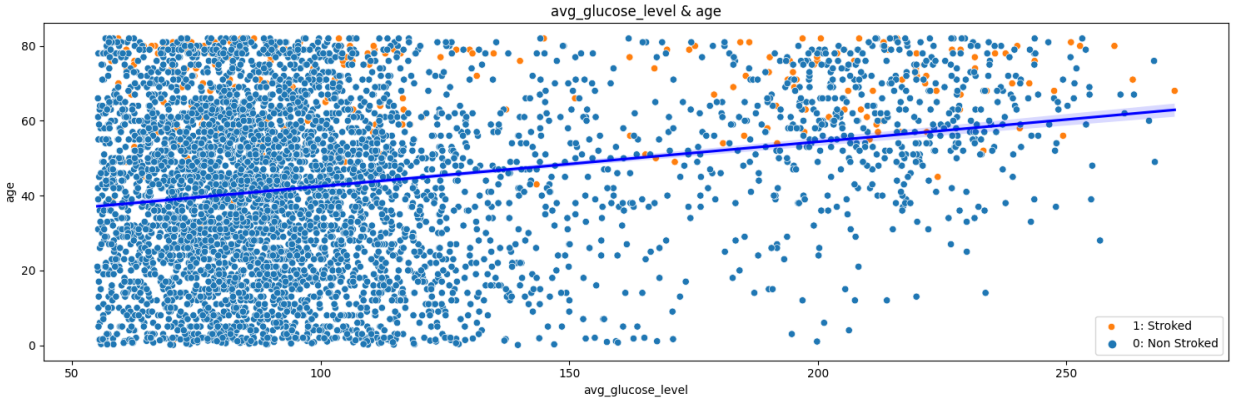
correlation, which means they are significant risk factors. Hypertension, with a similar correlation coefficient, which is consistent with known risk to stroke. Marital status unexpectedly shows to have a positive correlation, we didn't know whether there are some protential social or psychological influences on stroke risk. Smoking history is also positively correlated on cardiovascular health.

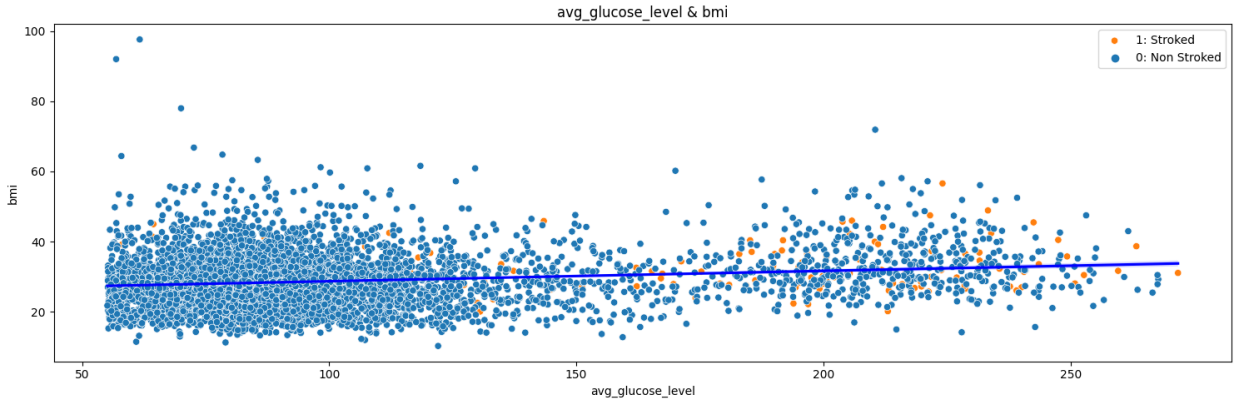


This below heatmap shows a comprehensive factor correlation, which represents the correlation coefficients between various variables in the dataset. The color intensity in the heatmap reflects the strength and direction of the correlation, with darker colors indicating stronger relationships. This figure is critical for identifying potential predictors of stroke. It can aid feature selection for predictive modeling by highlighting which variables are most closely associated with the occurrence of stroke. From the figure, it's clear that age has a notable positive correlation with the occurrence of stroke. This suggests that as people age, the risk of stroke increases. The average glucose level also shows a positive correlation with stroke, which is consistent with medical knowledge that elevated glucose levels can lead to vascular complications contributing to stroke risk. Additionally, the relationship between average glucose level and BMI is crucial, as it can indicate how body weight and glucose metabolism may jointly impact the risk of stroke.



Thus, we focused on the factors which are average glucose level, age and BMI. There are two figures below seem to describle two scatter plots with regression lines, one showing the relationship between average glucose level and age, and the other between average glucose level and BMI, each colored by the stroke outcome.





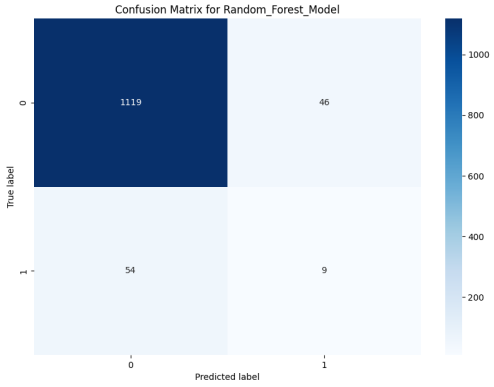
In the first figure about average glucose level and age, we could observe that as the average glucose level increases, the age of individuals also increases, which is indicated by the upward trend of the regression line. The color differentiation likely shows that higher average glucose levels and older age may correlate with a higher incidence of stroke, as indicated by the presence of more data points for stroke occurrences in these ranges. The second figure shows the relationship between average glucose levels and BMI. However, the regression line is less steep and almost close to flat, which might present a weaker relationship between glucose levels and BMI than with age. If we see clusters of stroke occurrences at higher glucose levels regardless of BMI, it could indicate that glucose levels are a more critical risk factor for stroke than BMI. These data visualizations could bring us revealing outcomes. If both age and average glucose levels show a clear positive trend with stroke occurrence but BMI does not, we would consider that the timely treatment to prevent stroke should prioritize glucose control and monitoring in older people instead of BMI.

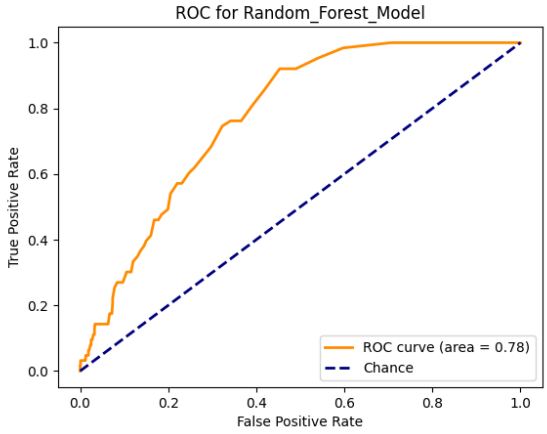
So far, we had implemented a series exploratory data analysis, which give us insights on the critical features of prredicting stroke risk. Since our main goal is to build a model that is both accurate and clinically relevant, providing healthcare professionals with a reliable tool for assessing stroke risk, we started to the feature scaling and model selection. Firstly, the dataset is split into two parts, around 75% for training to build the model and 25% for testing its predictions. This split is essential to validate the model's performance on unseen data. A significant challenge addressed next is the sample imbalance, which is the prevalent in medical datasets. We had only 4.9% for dignosised stroke smaples and 95.1% is taken up by unconfirmed stroke samples. To solve this issue to reduce the bias in the model towards the majority class, we used the Synthetic Minority Over-sampling Technique (SMOTE), which synthetically augments the minority class by generating new examples. This technique ensures that our predictive model is exposed to a balanced view of the stroke

occurrences, mitigating the risk of bias. Then, we implemented data normalization to scale the feature values and ensure that the range of our data values didn't influence the model's performance. This step is particularly important as it ensures that each feature contributes proportionately to the model's learning process, thereby enhancing the model's ability to learn patterns effectively and improving algorithm convergence during training. These preprocessing steps build a robust foundation for developing our stroke prediction model, so that the data is becoming representative, balanced, and scaled appropriately for the algorithm.

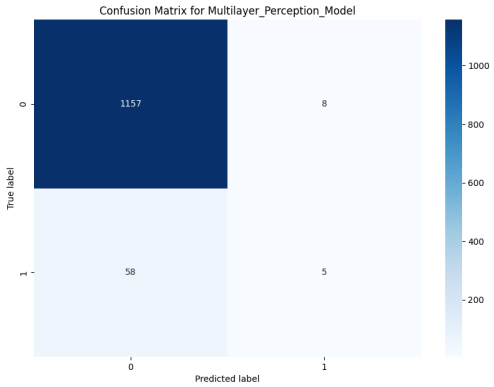
The evaluation of our models was thorough, utilizing both the confusion matrix and the ROC curve. The confusion matrix helped us understand the accuracy of our classifications, which reveals the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions. And the ROC curve is instrumental in assessing the performance of a binary classifier system, which provides insights into the balance between sensitivity and specificity. It plots the true positive rate against the false positive rate at various threshold settings. And the area under the curve (AUC) is a measure of the model's ability to distinguish between the two classes.

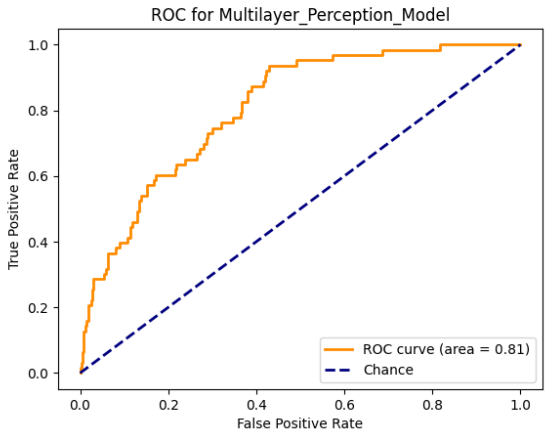
The first model we selected is Random Forest, after cross-validation we found optimal hyperparameters and utilized them to train and test the model. The below left figure is Random Forest model's confusion matrix. We see a high number of TN predictions, where the model correctly predicts the non-occurrence of strokes, and a relatively low number of TP predictions, indicating the model's challenge in correctly identifying actual stroke cases. This discrepancy is a common issue in medical prediction models, especially when dealing with imbalanced datasets where the occurrence of one class (stroke) is much less frequent than the other (non-stroke). The below right figure is Random Forest model's ROC curve. An AUC of 0.78 suggests that the model has a reasonable distinction capacity, although there is room for improvement, especially in the context of medical predictions where higher accuracy is necessary. In summary, the Random Forest model shows promising results but also reflects the challenge of predicting relatively rare cases such as strokes. The use of techniques like SMOTE for oversampling and rigorous hyperparameter tuning through grid search is evidenced to enhance the model's performance. However, the model's difficulty in accurately identifying stroke cases, we could look for further refinement, through advanced feature engineering, sample scaling strategies, or different algorithms to improve the predictive accuracy for the minority class. This level of detail in analysis ensures that the model's predictive power aligns closely with the critical healthcare objective of accurately identifying stroke risk.





The second model we selected is Mutilayer Perception, after cross-validation we found optimal hyperparameters and utilized them to train and test the model. The below left figure is Mutilayer Perception model's confusion matrix. Simmilar to the Random Forest model results, Mutilayer Perception model also has a high number of TN predictions, which means it generates many false positives and failed to catch numerous stroke cases, leading to a lower recall for these critical cases and therefore a greater chance of overlooking actual stroke patients. Eventhough it ranks higher in accuracy and AUC, but it couldn't detect stroke cases. This is a major drawback for medical diagnosis where missing a true case can have serious consequences. From these results, we would say that neither model is effective for stroke prediction, but Mutilayer Perception could be a choice if there is no other models to compare.





Conclusion

The main goal of the project is to identify key factors influencing stroke risk. By developing a predictive model based on these factors, we can provide doctors with a powerful tool for assessing stroke risk more accurately. This not only aids in medical professionals in risk assessment, but also offers patients timely medical advice, providing actionable insights.

According to the exploratory data analysis summaries, we found the correlated factors between various features and occurrence of stroke. We identified key factors influencing stroke risk such as numerical variables age, average glucose level and BMI. We made a lot of relevant speculations and thoughts based on the data and visualization results. For instance, higher average glucose levels and older age may correlate with a higher incidence of stroke. These analysises can help us to determine whether features are useful.

For stroke dignosis classifier, we compared two classification model Random Forest and Multilayer Perception. From the results they got, the Random Forest classifier with an overall accuracy of 91.85%, while the Multilayer Perceptron classifier indicates a higher overall accuracy of 94.63%. The second model performs slightly better in terms of overall accuracy. As we mentioned before, both models perform well on the majority class, with high precision, recall, and F1-scores. This is expected due to the class imbalance. This is a challenge for us when using machine learning and the algorithm can lead to biased models that are better at predicting the majority class at the expense of the minority class. Altogether, the Multilayer Perceptron model has a better balance between precision and recall for the minority class, as indicated by the higher precision. Therefore, for this case to predict stroke risk, using Multilayer Perceptron would be better than Random Forest, eventhough both of them have low recall for class 1 (stroke), indicating many false negatives (non-stroke).

Although our current results do not meet our expectations, we believe that both models would benefit from techniques to handle the class imbalance, such as oversampling the minority class or using anomaly detection methods, we will try different techniques in the future. This is what we would continue to try next.

Supplementary materials

Feature	Description
id	unique identifier
gender	“Male”, “Female” or “Other”
age	age of the patient
hypertension	0 if the patient doesn’t have hypertension, 1 if they do
heart_disease	0 if the patient doesn’t have any heart diseases, 1 if they do
ever_married	“No” or “Yes”
work_type	“children”, “Govt_job”, “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

Data Preprocessing

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import warnings
warnings.simplefilter("ignore")
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.tree import DecisionTreeClassifier, export_graphviz
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeRegressor
from graphviz import Source
from sklearn.metrics import roc_curve, auc
from imblearn.over_sampling import SMOTE
from collections import Counter
```

```
df = pd.read_csv("healthcare-dataset-stroke-data.csv")
df.drop(columns=['id'], inplace=True)
#print(df.shape)
#print(df.describe())
```

Handle Missing Value

```
df.isna().sum()
```

```
dt_bmi_pipe = Pipeline(steps=[
    ('scale', StandardScaler()),
    ('lr', DecisionTreeRegressor(random_state=42))
])
temp_X = df[['age','gender','bmi']].copy()
temp_X.gender = temp_X.gender.replace({
    'Male':0,'Female':1,'Other':-1
}).astype(np.uint8)
missing = temp_X[temp_X.bmi.isna()]
temp_X = temp_X[~temp_X.bmi.isna()]
temp_Y = temp_X.pop('bmi')
dt_bmi_pipe.fit(temp_X,temp_Y)
predicted_bmi = pd.Series(dt_bmi_pipe.predict(
```

```
    missing[['age','gender']]),index=missing.index)
df.loc[missing.index,'bmi'] = predicted_bmi
df.isna().sum()
```

Encode Categorical Data

```
numeric_columns = ['age', 'bmi', 'avg_glucose_level']
categorical_columns = ['gender', 'hypertension', 'heart_disease',
                       'ever_married', 'work_type', 'Residence_type',
                       'smoking_status', 'stroke']
# use dictionary to save column name as key, save categories as value
object_categorical_columns_dict = {}
for col in df.columns:
    if df[col].dtype == 'object':
        object_categorical_columns_dict[col]=df[col].value_counts().index.to_list()
#print(object_categorical_columns_dict)
```

Method 1: One-hot Encoding

```
df.info()
```

```
# one hot encoding for multiple categories
df = pd.get_dummies(df, columns=object_categorical_columns_dict.keys())
df.info()
```

Method 2: Label Encoding

```
encoded_values = {}
temp_dict = {}
print('_'*45)
for col, values in object_categorical_columns_dict.items():
    print(f'column: {col}')
```

```
    for index, val in enumerate(values):
        temp_dict[val] = index
        print(f'{val}: {index}')
    encoded_values[col] = temp_dict
    df[col] = df[col].replace(temp_dict)
    print('_'*45)
```

Explore and Visualise Data

stroke: 1 if the patient had a stroke or 0 if not(FEDESORIANO 2023)

Numeric Columns: Age, bmi, avg_glucose_level

```
i = 0
fig, ax = plt.subplots(3, 3, figsize=(18, 8))
plt.subplots_adjust(hspace = 0.5)
for num_col in numeric_columns :
    sns.kdeplot(x=num_col, hue='stroke', data=df, multiple='stack', ax=ax[i,0])
    sns.boxplot(x=num_col, data=df, ax=ax[i, 1])
    sns.scatterplot(x=num_col, y='stroke', data=df, ax=ax[i, 2])
    i+=1
plt.close()
```

Those who have had a stroke are in: age in range 40 to 85, bmi in range 20 to 40, glocuse level in range 50 to 130

```
# Correlation for the target variable (stroke)
plt.figure(figsize = (15,5))
sns.heatmap(df.corr()[['stroke']].sort_values(
    by='stroke', ascending=False), annot = True, cmap = 'Blues')
plt.title('Target Correlation',fontsize=16)
plt.tight_layout()
```

```
plt.savefig(f'Target_Correlation.png')
plt.close()
```

```
# Correlation between each factor
corr = df.corr()
plt.figure(figsize = (18,10))
sns.heatmap(corr, cmap = 'Blues', annot = True)
plt.title("Factor Correlation", fontsize=16)
plt.tight_layout()
plt.savefig(f'Factor_Correlation.png')
plt.close()
```

```
plt.figure(figsize = (15,5))
sns.scatterplot(data=df, x='avg_glucose_level',y='age',hue='stroke')
sns.regplot(data=df, x='avg_glucose_level', y='age',
            scatter=False, ci=95, line_kws={"color": "blue"})
plt.xlabel('avg_glucose_level')
plt.ylabel('age')
plt.title('avg_glucose_level & age')
plt.legend(labels=['1: Stroked', '0: Non Stroked'])
plt.tight_layout()
plt.savefig(f'avg_age.png')
plt.close()
```

```
plt.figure(figsize = (15,5))
sns.scatterplot(data=df, x='avg_glucose_level',y='bmi',hue='stroke')
sns.regplot(data=df, x='avg_glucose_level', y='bmi',
            scatter=False, ci=95, line_kws={"color": "blue"})
plt.xlabel('avg_glucose_level')
plt.ylabel('bmi')
plt.title('avg_glucose_level & bmi')
plt.legend(labels=['1: Stroked', '0: Non Stroked'])
```

```
plt.tight_layout()
plt.savefig(f'avg_bmi.png')
plt.close()
```

Feature Scaling and Upscaling

Data Splitting

split the data into training set and testing set

```
X = df.drop(columns='stroke')
y = df['stroke']
X.columns
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42)
```

Handle Data Imbalance

```
x = df['stroke'].value_counts()
explode = [0, 0.1]
fig, ax = plt.subplots(figsize=(4, 4),subplot_kw=dict(aspect="equal"))
plt.pie(x,explode=explode,autopct='%1.1f%%',textprops=dict(color="w",size=10))
plt.legend(labels = ['0: Non Stroke', '1: Stroke'])
plt.close()
```

```
sm = SMOTE(random_state=42)
X_train_oversampled, y_train_oversampled = sm.fit_resample(X_train, y_train)
print(f'Original class distribution: {Counter(y_train)}')
print(f'Oversampled class distribution: {Counter(y_train_oversampled)}')
X_train = X_train_oversampled
y_train = y_train_oversampled
```

```
Original class distribution: Counter({0: 3663, 1: 169})
Oversampled class distribution: Counter({0: 3663, 1: 3663})
```

Normalization

```
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
X_train = pd.DataFrame(X_train_scaled, columns=X_train.columns)
X_test = pd.DataFrame(X_test_scaled, columns=X_test.columns)
#X_train.describe()
```

Data Visualization Functions

```
def cm(y_test, y_predict, modelName):
    cm = confusion_matrix(y_test, y_predict)
    plt.figure(figsize=(10, 7))
    sns.heatmap(cm, annot=True, fmt='g', cmap='Blues')
    plt.xlabel('Predicted label')
    plt.ylabel('True label')
    plt.title(f'Confusion Matrix for {modelName}')
    plt.savefig(f'{modelName}_cm.png')
    plt.close()

def roc(y_test, y_pred_probs, modelName):
    fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
    roc_auc = auc(fpr, tpr)
    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--', label='Chance')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
```

```
    plt.title(f'ROC for {modelName}')
    plt.legend(loc="lower right")
    plt.savefig(f'{modelName}_roc.png')
    plt.close()
```

Model Selection

Model 1: Random Forest

Cross Validation

```
# set paremeter grid, create cross validation by grid search
param_gs = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'max_depth': [3,5, None],
    'n_estimators': [50, 100]
}
cv_rf = RandomForestClassifier()
cv_gs_rf = GridSearchCV(estimator=cv_rf, cv=5, param_grid=param_gs)
cv_gs_rf.fit(X_train, y_train)
print("Optimal tuning parameters: ", cv_gs_rf.best_params_)
print("Optimal cross-validation score: ", cv_gs_rf.best_score_)
```

Training and Prediction

```
criterion = cv_gs_rf.best_params_['criterion']
max_depth = cv_gs_rf.best_params_['max_depth']
n_estimators = cv_gs_rf.best_params_['n_estimators']
rf = RandomForestClassifier(
    criterion=criterion, max_depth=max_depth, n_estimators=n_estimators)
rf.fit(X_train, y_train)
rf_score = round(rf.score(X_test, y_test), 5)
```

```
y_predict = rf.predict(X_test)
report = classification_report(y_test, y_predict)
print(f'Accuracy: {rf_score}')
print(report)
```

Confusion Matrix

```
modelName = 'Random_Forest_Model'
cm(y_test, y_predict, modelName)
```

ROC

```
y_pred_prob = rf.predict_proba(X_test)[:, 1]
roc(y_test, y_pred_prob, modelName)
```

```
random_forest_model = RandomForestClassifier()
random_forest_model.fit(X, y)
dt_fit = random_forest_model.estimators_[0]
dot_data = export_graphviz(dt_fit, out_file=None, feature_names=X.columns,
                           class_names=['0: Non Stroke', '1: Stroke'],
                           filled=True, rounded=True, special_characters=True)
graph = Source(dot_data)
graph.render(f"{modelName}_graph", format='png', cleanup=True)
```

Model 2: Multilayer Perception

Cross Validation

```
params_gs = {
    'hidden_layer_sizes':[8,16,32],
    'activation':['identity', 'logistic', 'tanh', 'relu'],
    'solver': ['lbfgs', 'sgd', 'adam'],
```

```
    'learning_rate':['constant', 'invscaling', 'adaptive'],
    'max_iter': [100,200],'warm_start':[True]
}
cv_mlp = MLPClassifier()
cv_gs_mlp = GridSearchCV(estimator=cv_mlp, param_grid=params_gs,
                         cv=5, scoring='accuracy')
cv_gs_mlp.fit(X_train, y_train)
print("Optimal tuning parameters: ", cv_gs_mlp.best_params_)
print("Optimal cross-validation score: ", cv_gs_mlp.best_score_)
```

Training and Prediction

```
mlp = MLPClassifier(**cv_gs_mlp.best_params_)
mlp.fit(X_train, y_train)
y_predict = mlp.predict(X_test)
report = classification_report(y_test, y_predict)
mlp_score = mlp.score(X_test, y_test)
print(f'Accuracy: {mlp_score}')
print(report)
```

Confusion Matrix

```
modelName = 'Multilayer_Perception_Model'
cm(y_test, y_predict, modelName)
```

ROC

```
y_pred_prob = mlp.predict_proba(X_test)[:, 1]
roc(y_test, y_pred_prob, modelName)
```

Reference

FEDESORIANO. 2023. *Stroke Prediction Dataset*. https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset/data.

SHARMA, TANMAY. 2023. *Deep Learning[weight Balancing and Oversampling]*. https://www.kaggle.com/code/tanmysharma123/deep-learning-weight-balancing-and-oversampling#Missing-Values.

Uma Venugopal, Ashray Agur. 2023. *Stroke Prediction*. https://www.kaggle.com/datasets/teamincribo/stroke-prediction.