In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, r
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [2]:

```
import os
os.chdir(r"C:\Users\mikep\Documents\WGU\D214 Capstone")
df = pd.read_csv('patient_data.csv')
```

In [4]:

```
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40910 entries, 0 to 40909
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sex                40907 non-null  float64
 1   age                40910 non-null  float64
 2   hypertension       40910 non-null  int64  
 3   heart_disease      40910 non-null  int64  
 4   ever_married       40910 non-null  int64  
 5   work_type          40910 non-null  int64  
 6   Residence_type     40910 non-null  int64  
 7   avg_glucose_level  40910 non-null  float64
 8   bmi                40910 non-null  float64
 9   smoking_status     40910 non-null  int64  
 10  stroke             40910 non-null  int64  
dtypes: float64(4), int64(7)
memory usage: 3.4 MB
```

Out[4]:

	sex	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level
0	1.0	63.0	0	1	1	4	1	228.69
1	1.0	42.0	0	1	1	4	0	105.92
2	0.0	61.0	0	0	1	4	1	171.23
3	1.0	41.0	1	0	1	3	0	174.12
4	1.0	85.0	0	0	1	4	1	186.21

In [5]:

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

Out[5]:

```
sex                  3
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 [6]:

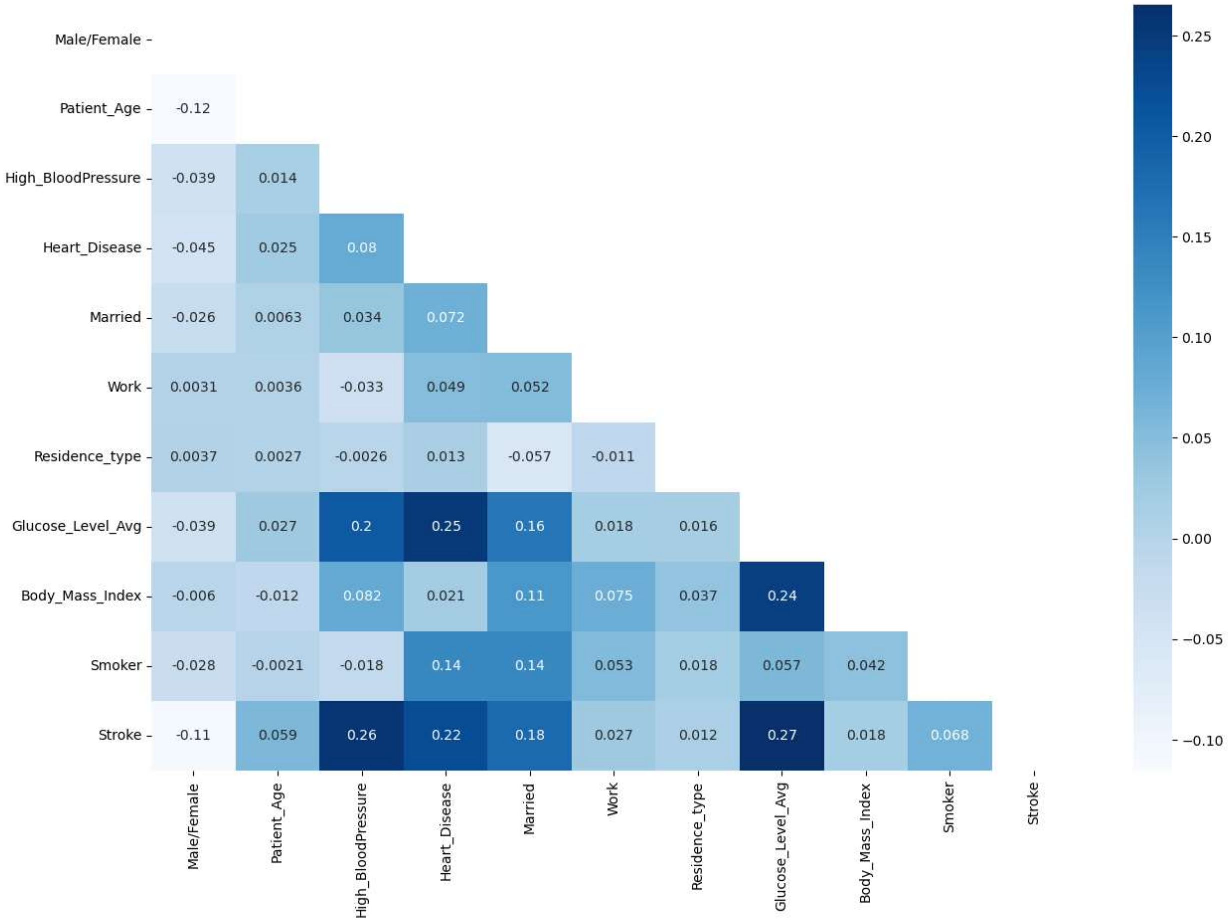
```
df = df.dropna()
```

In [7]:

```
df = df.rename(columns={'sex': 'Male/Female', 'age': 'Patient_Age', 'hypertension': 'H
```

In [8]:

```
plt.figure(figsize=(15,10))
cormask=np.triu(df.corr()) #masks redundnat half of heatmap
sns.heatmap(df.corr(), mask=cormask, cmap="Blues", annot=True)
plt.show()
```



In [9]:

```
X = df[[col for col in df.columns if col != 'Stroke']]
y = df['Stroke']
```

In [10]:

```
#########################################################################################
```

In [11]:

```
rf = RandomForestClassifier()
```

In [12]:

```
param_grid = {'n_estimators': [100, 200, 500],
              'max_depth': [2, 4, 6, 8],
              'min_samples_split': [2, 5, 10],
              'min_samples_leaf': [1, 2, 4],
              'max_features': ['sqrt', 'log2'],
              'bootstrap': [True, False]}
```

In [13]:

```
RSCV_rf = RandomizedSearchCV(estimator=rf, param_distributions=param_grid, cv=5, n_ite
RSCV_rf.fit(X, y)
```

Out[13]:

```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(), n_iter=100,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [2, 4, 6, 8],
                                        'max_features': ['sqrt', 'log2'],
                                        'min_samples_leaf': [1, 2, 4],
                                        'min_samples_split': [2, 5, 10],
                                        'n_estimators': [100, 200, 500]},
                   scoring='f1')
```

In [14]:

```
# Print the best hyperparameters and validation score
print('Best hyperparameters:', RSCV_rf.best_params_)
print('F-score:', RSCV_rf.best_score_)
```

```
Best hyperparameters: {'n_estimators': 200, 'min_samples_split': 2, 'min_samples_lea
f': 2, 'max_features': 'sqrt', 'max_depth': 8, 'bootstrap': False}
F-score: 0.7478900565784681
```

In [15]:

```
param_grid2 = {'n_estimators': [300, 450, 550],
              'max_depth': [6, 8, 10],
              'min_samples_split': [2, 3, 5],
              'min_samples_leaf': [2, 3, 4],
              #'max_features': [sqrt],
              'bootstrap': [False]}
```

In [16]:

```
RSCV_rf2 = RandomizedSearchCV(estimator=rf, param_distributions=param_grid2, cv=5, n_i
RSCV_rf2.fit(X, y)
```

Out[16]:

```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(), n_iter=80,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [False],
                                        'max_depth': [6, 8, 10],
                                        'min_samples_leaf': [2, 3, 4],
                                        'min_samples_split': [2, 3, 5],
                                        'n_estimators': [300, 450, 550]},
                   scoring='f1')
```

In [17]:

```
print('Best hyperparameters:', RSCV_rf2.best_params_)
print('F-score:', RSCV_rf2.best_score_)
```

```
Best hyperparameters: {'n_estimators': 300, 'min_samples_split': 5, 'min_samples_lea
f': 2, 'max_depth': 10, 'bootstrap': False}
F-score: 0.8574557776371202
```

In [18]:

```
param_grid3 = {'n_estimators': [250, 375, 500],
              'max_depth': [8, 10, 12],
              'min_samples_split': [2, 3],
              'min_samples_leaf': [2, 3, 4],
              #'max_features': ['sqrt', 'log2'],
              'bootstrap': [False]}
```

In [19]:

```
RSCV_rf3 = RandomizedSearchCV(estimator=rf, param_distributions=param_grid3, cv=5, n_i
RSCV_rf3.fit(X, y)
```

Out[19]:

```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(), n_iter=50,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [False],
                                        'max_depth': [8, 10, 12],
                                        'min_samples_leaf': [2, 3, 4],
                                        'min_samples_split': [2, 3],
                                        'n_estimators': [250, 375, 500]},
                   scoring='f1')
```

In [20]:

```
print('Best hyperparameters:', RSCV_rf3.best_params_)
print('F-score:', RSCV_rf3.best_score_)
```

```
Best hyperparameters: {'n_estimators': 375, 'min_samples_split': 2, 'min_samples_lea
f': 4, 'max_depth': 12, 'bootstrap': False}
F-score: 0.9378982769389582
```

In [21]:

```
param_grid4 = {'n_estimators': [250, 275, 300],
              'max_depth': [10, 13, 15, 20],
              'min_samples_split': [2, 3],
              'min_samples_leaf': [3, 4],
              #'max_features': ['sqrt', 'log2'],
              'bootstrap': [False]}
```

In [22]:

```
RSCV_rf4 = RandomizedSearchCV(estimator=rf, param_distributions=param_grid4, cv=5, n_i
RSCV_rf4.fit(X, y)
```

```
C:\Users\mikep\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:292: Us
erWarning: The total space of parameters 48 is smaller than n_iter=50. Running 48 ite
rations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
```

Out[22]:

```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(), n_iter=50,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [False],
                                        'max_depth': [10, 13, 15, 20],
                                        'min_samples_leaf': [3, 4],
                                        'min_samples_split': [2, 3],
                                        'n_estimators': [250, 275, 300]},
                   scoring='f1')
```

In [23]:

```
print('Best hyperparameters:', RSCV_rf4.best_params_)
print('F-score:', RSCV_rf4.best_score_)
```

```
Best hyperparameters: {'n_estimators': 250, 'min_samples_split': 2, 'min_samples_lea
f': 3, 'max_depth': 20, 'bootstrap': False}
F-score: 0.9956228495310186
```

In [24]:

```
#####################################################################################
```

In [25]:

```
from sklearn.model_selection import GridSearchCV
param_grid5 = {'n_estimators': [275, 300],
              'max_depth': [18, 22, 25],
              'min_samples_split': [3, 4],
              'min_samples_leaf': [2, 3]},
              #'max_features': ['sqrt', 'log2'],
              #'bootstrap': [True, False]}
```

In [26]:

```
GSCV_rf = GridSearchCV(estimator=rf, param_grid=param_grid5, cv= 3)
GSCV_rf.fit(X, y)
GSCV_rf.best_params_
```

Out[26]:

```
{'max_depth': 22,
 'min_samples_leaf': 2,
 'min_samples_split': 3,
 'n_estimators': 275}
```

In [27]:

```
print('Best hyperparameters:', GSCV_rf.best_params_)
print('F-score:', GSCV_rf.best_score_)
```

```
Best hyperparameters: {'max_depth': 22, 'min_samples_leaf': 2, 'min_samples_split':
3, 'n_estimators': 275}
F-score: 0.9952331506414224
```

In [28]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, stratify=y,
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(28634, 10) (28634,)
(12273, 10) (12273,)
```

In [29]:

```
# Train the model with the best hyperparameters
bestmodel = RSCV_rf4.best_estimator_
bestmodel.fit(X_train, y_train)
train_acc = bestmodel.score(X_train, y_train)
test_acc = bestmodel.score(X_test, y_test)
print("Train accuracy:", train_acc)
print("Test accuracy:", test_acc)
```

```
Train accuracy: 0.9997206118600266
Test accuracy: 0.9954371384339608
```

In [30]:

```
y_proba = bestmodel.predict_proba(X_test)[:, 1]
roc_auc = roc_auc_score(y_test, y_proba)
print("ROC AUC:", roc_auc)

y_pred = bestmodel.predict(X_test)
f1score = f1_score(y_test, y_pred)
print("F1-score:", f1score)
```

```
ROC AUC: 0.9999946357387796
F1-score: 0.9954589685371392
```

In [31]:

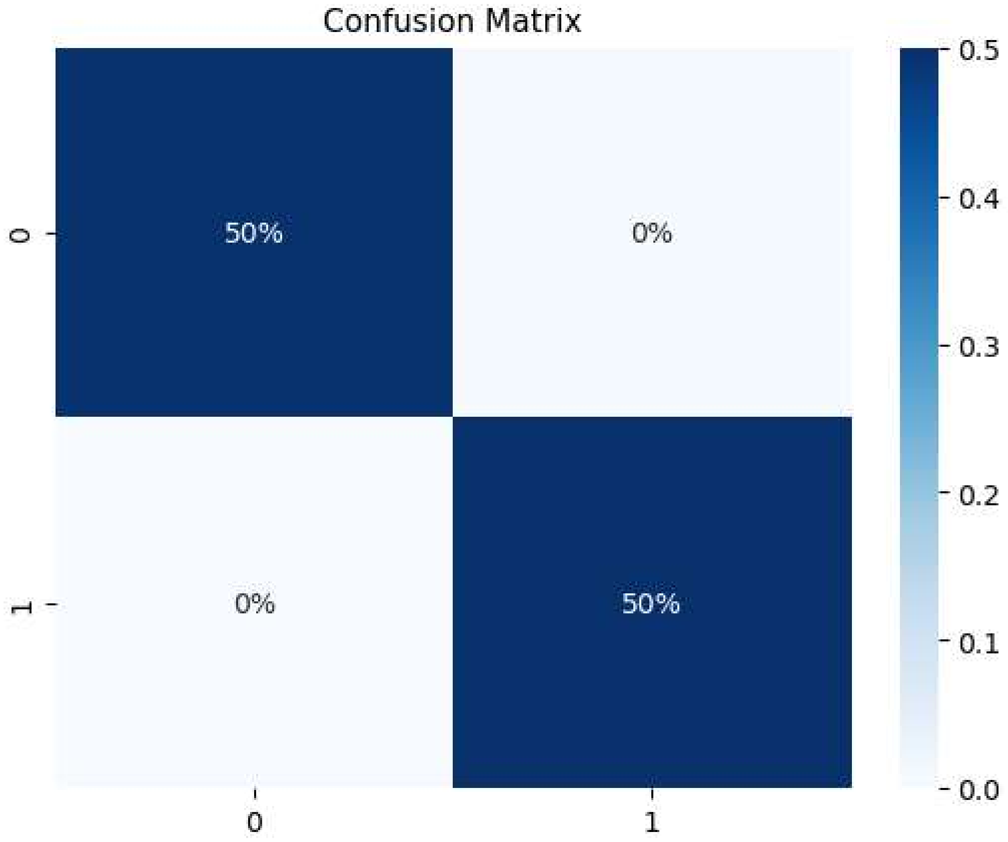
```
#####################################################################################
```

In [32]:

```
confmatrix = confusion_matrix(y_test, y_pred)
sns.heatmap(confmatrix/np.sum(confmatrix), annot=True, fmt='.00%', cmap='Blues')
plt.title("Confusion Matrix", fontsize =11)
```

Out[32]:

```
Text(0.5, 1.0, 'Confusion Matrix')
```



In [ ]: