Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

# ITA5007 – Data Mining and Business Intelligence

J-Component

Faculty Name: Dr. Brijendra Singh

## Project Title: Heart Disease Prediction using Data Mining techniques


Team Members:

Shubham Jha [22MCA0398]

Indranil Sarkar [22MCA0343]

Ankit Kumar [22MCA0391]

## ABSTRACT

Heart Disease is one of the deadliest diseases in the world with very high mortality rates. It has been predicted that around 12 million people die every year due to cardiovascular diseases. Early prediction for such a deadly disease is thus very important as it leads to saving of countless lives with the help of early treatment thus reducing the chances of complications. It may be caused due to many factors like smoking habits, hypertension, cholesterol levels, irregular Blood Pressure, unusual heart rate, etc. The cure for such a deadly disease is not appropriately available especially when it grows in complication and requires lifetime of treatment and careful monitoring throughout. This paper aims at predicting the possibility of a heart stroke based on a recent dataset using various data mining techniques. We will be using PCA as a preprocessing method and various ML algorithms such as Logistic Regression, SVM and Random Forest, KNN and K-fold cross validation techniques. The results are then compared and analyzed.

**Keywords:** Heart Disease, PCA, Logistic Regression, Random Forest, KNN

## INTRODUCTION

There have been huge advancements in the medical field in the last few decades as the medical science as well as the infrastructure to study them grew in scale. But there are still some conditions that don't have any proper treatment, specially in the later stages. Heart disease is one such disease with no proper medical treatment. Medical professionals usually identify healthy lifestyle as a treatment to prevent cardiovascular diseases such as heart stroke, cardiac arrest, etc from happening. Heart is a major organ of our body containing a network of blood vessels that pumps blood throughout our body. Its major function is to maintain proper heart rate and blood pressure. It consists of chambers to store blood and valves to allows blood flow between chambers or towards the body via the blood vessels.

Heart Stroke is a situation that occurs when blood stops flowing to any part of the brain, thus damaging the brain cells. This is generally caused by blockage of the blood vessels to the brain due to some reasons like high blood pressure, smoking, diabetes, etc. With the use of various data mining techniques in this paper, we aim to predict the early occurrence of the stroke with good accuracy.

# LITERATURE REVIEW

| Ref No. | Title of the Paper | Algorithms Used | Dataset used | Result/ Conclusion | Limitations/ Scope |
|---|---|---|---|---|---|
| [1] | Data Science and its Application in Heart Disease Prediction | Naïve Bayes, ANN, SVM and Hybrid Naïve Bayes, SVM, and ANN. | From Kaggle. With 309 samples and 14 features. | Hybrid models are Effective and providing the highest accuracy, specificity, and sensitivity with 82.11% and 91.47 % respectively. | This study provides a fresh approach for the study of how these ideas might be used to smart devices and data science to transform the detection and treatment of cardiac disease. |
| [2] | Heart Disease Prediction Algorithm Based on Ensemble Learning | hybrid gradient boosting decision tree with logistic regression (HGBDTLR) | UCI Cleveland dataset | In the Cleveland heart disease data set, the HGBDTLR algorithm's prediction accuracy can go as high as 91.8%. | One of the key factors affecting the classifier's performance is the restriction on feature selection. Dataset is very small and can't be used for real time applications |
| [3] | Prediction of Heart Disease Using Machine Learning. | neural networks | Cleveland dataset from UCI library. | The Heart Disease Prediction System, which employs the machine learning algorithm MLP, delivers its customers a prediction result that indicates the user's state as it relates to CAD. | With the use of modern technologies like machine learning, fuzzy logics, image processing, and many others, comparable prediction systems can be created for a variety of other deadly or chronic conditions like Cancer, Diabetes, etc. |

| | | | | | |
|---|---|---|---|---|---|
| [4] | Heart Disease Prediction using Hybrid machine Learning Model | Random forest regression, Decision tree, Hybrid Model | Cleveland dataset from uci.edu | Decision tree: 79% Random forest: 81% Hybrid (Decision tree + Random forest):88% | Deep learning algorithms can be used to get better outcomes. Multi-class problems can be used to predict the levels of diseases. |
| [5] | Efficient Heart Disease Prediction System using Decision Tree | Decision tree | Cleveland dataset | Total accuracy: 86.75% Partition 1: 86.3% Partition 2: 87.2% | More advanced model can be used for better accuracy and precision. |
| [6] | Heart disease prediction based on random forest and LSTM | Random forest, LSTM, KNN, DNN | Heart disease dataset from Kaggle | Random tree—LSTM: 0.8729508 LSTM : 0.84562844 Random tree—DNN: 0.8621554 DNN: 0.8278689 Random tree—KNN: 0.8461233 KNN: 0.8224044 | Used basic level of optimization for the model . More advanced and professional level tuning can be done to acquire better results. |
| [7] | Prediction of Heart Disease using DNN | DNN | Cleveland Dataset | Accuracy: 81.9%, 85% with AdaGrad optimizer | Generative models can be used to enhance the size of the dataset. |
| [8] | Classification technique for Heart Disease Prediction in Data Mining | KNN | Cleveland Dataset | Accuray: 83% Less execution time comparatively | In future, more classifiers can be combined to form hybrid classifier. |
| [9] | Heart Disease prediction using MLP and LSTM models | MLP and LSTM | Heart UCI dataset | MLP: 89.18% LSTM: 96.5% | Geographic limitations as the dataset may vary when other regions. |

# OBJECTIVES

- Allow for Early Detection of Heart Disease
  With early detection, necessary preventive measures can be taken to prevent any further complications of the heart disease.

- Use ML feature selection methods for less computations
  We will be using PCA(Principal Component Analysis) dimensionality reduction method to reduce the number of features but still conserving the essence of the data.

- Apply ML techniques for prediction
  Using ML techniques like Logistic Regression, KNN and Decision Tree, we will be predicting if the patient, based on their data, may have heart stroke or not.

- Identify relationship between important factors
  Use various EDA techniques to find correlation between various factors affecting heart stroke and how and up to what scale they influence each other.

- Support further Medical Research
  Advocate for preventive measures like changes in lifestyle and dietary habits and encourage necessary medications. With the current heart disease prediction system, allow for discovery for new risk factors, treatment approaches and further study and advancements in cardiovascular health.

# DATASET

The dataset for the Heart Disease prediction was downloaded from Kaggle.com.

Link - https://www.kaggle.com/datasets/mirzahasnine/heart-disease-dataset

Code - https://colab.research.google.com/drive/1ewkMQ6A5pjkMtEpIYZz_OX2b5PbjUXla

The dataset contains information on patients with heart stroke. The dataset is very recent having been uploaded in the month of December, 2022.

The dataset contains 4238 samples and 16 features including the binary target variable. The features are discussed below:

1. Gender: Gender of the person(Male/Female)
2. Age: Age of the person.
3. Education: Education Level of the person(Primary School, graduate, Post graduate, Uneducated)
4. currentSmoker: If person smokes currently.
5. cigsPerDay: Number of times, person smokes in a day.
6. BPMeds: Binary variable suggesting if a person has medications for BP.
7. prevalentStroke: Binary variable suggesting if a person has had stroke previously.
8. prevalantHyp: Binary variable suggesting if a person has preexisting Hypertension condition.
9. Diabetes: Binary variable suggesting if a person has diabetes.
10. totChol: Cholesterol level of a person.
11. sysBP: Systolic blood pressure of a person.
12. diaBP: Diastolic blood person of a person.
13. BMI: Body Mass Index(BMI) of a person.
14. heartRate: Heart rate of a person.
15. Glucose: Glucose level of a person.
16. Heart_stroke: Binary variable suggesting if a person had stroke.

# METHODOLOGY

## I. PREPROCESSING

The dataset is passed through number of preprocessing tests to make it ready for training and testing a model. These include handling cases for null values in our samples. This can be achieved by filling the numerical values with the median of the remaining values. For a textual value, we can use the mode of the remaining values. Also, since textual values is difficult to work with, we map it into a numerical value using Label Encoder.

## II. EXPLORATRY DATA ANALYSIS

Visualization charts like heat map, scatter plot and histograms have been drawn to further understand the data in a visually effective manner to get further insights into the data and comment any prior analysis.

## III. DIMENSIONALITY REDUCTION

Principal Component Analysis (PCA) is used to reduce the number of dimensions(features) of the dataset from 15 to 3 while preserving 95% of the variability in the data. This helps hugely as the number of computations to be done is very much reduced while the dataset still has the same meaning.

## IV. MACHINE LEARNING

We have applied 3 different ML techniques discussed below:

1. Logistic Regression: It is a statistical model that models the probability of an event taking place by having log-odds for the event be a linear combination of one or more independent variables. This linear combination is then passed through a sigmoid function to give the class probability.

$$f(x) = \frac{1}{1+e^{-(x)}}$$

2. K-Nearest Neighbor (KNN): It is a non-parametric supervised learning classifier, which uses similarity to make predictions about the grouping of an individual data point. By default, it averages the results of its 5 neighbors and gives the prediction with majority count as its outcome. It uses Euclidean distance to calculate its nearest neighbors as:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

3. Random Forest: Decision Tree is a tree-based classifier where leaf nodes represent a particular class. The tree is traversed via some set of decision rules for a new node and a class is set for that particular record when the leaf node is reached. Random Forest is a classifier that contains 'n' number of decision trees that is built upon the subset of the original dataset. Higher number of trees leads to higher accuracy and reduces overfitting.

## V. COMPARISON

The performance of these algorithms are compared using the accuracy metric and the best performing algorithm is highlighted.

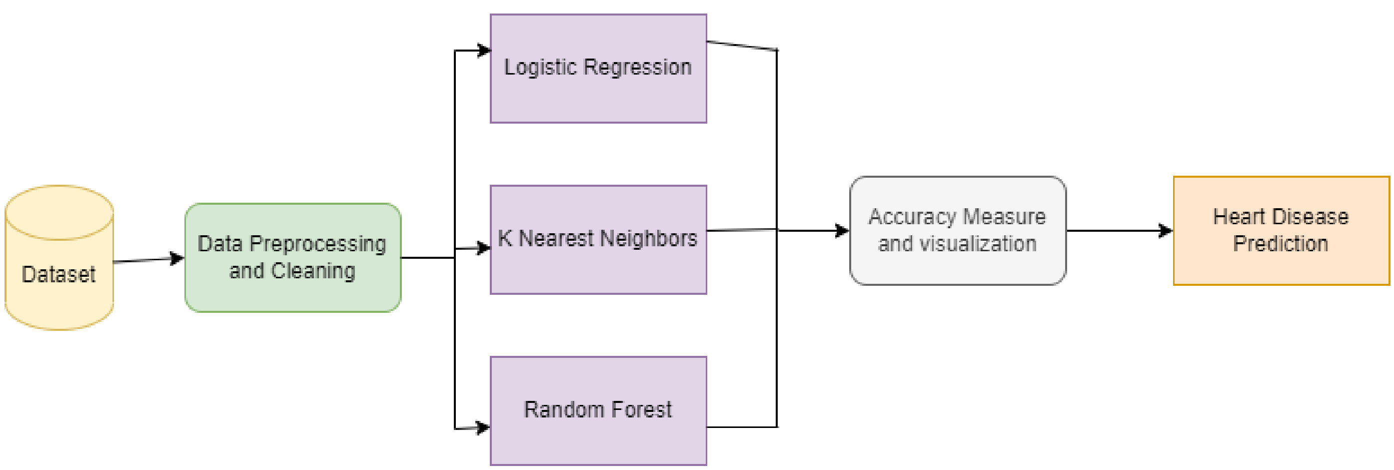



*Fig: Architecture Design*

# RESULTS

The dataset was split into training and testing data in the ratio 70:30. The training data was used to train all of the models and were individually tested on the testing dataset. The models are then compared based on the accuracy performance metrics. The results obtained from the applied ML models are as below:

| **Model** | **Accuracy** |
|---|---|
| Logistic Regression | 86 % |
| Logistic Regression with PCA | 83.72 % |
| KNN | 84.67 % |
| KNN with PCA | 82.3 % |
| Random Forest | 86.33 % |
| Random Forest with PCA | 84.51 % |

As observed from the above table, the Random Forest algorithm performed the best compared to the other algorithms with an accuracy of 86.33 %. Logistic Regression also performed well comparatively while KNN had the least favorable accuracy.

For the Random Forest model, we have set the hyperparameter, n_estimators to 100 meaning that same amount of trees will make up the forest for the model. For the Logistic Regression model we have used liblinear solver for optimization of the loss function. For KNN, we have set the number of neighbors parameter to 5.

We can also observe from the results table that initially our dataset had 15 independent features which is reduced to 3 using the Principal Component Analysis (PCA) while retaining 95% variation of the original dataset. The resultant accuracy is on par with the accuracy of the model with original dataset. This highlights the power of PCA as preprocessing step for dimensionality reduction and feature extraction while reducing the computational requirements and simplifying the model building process.

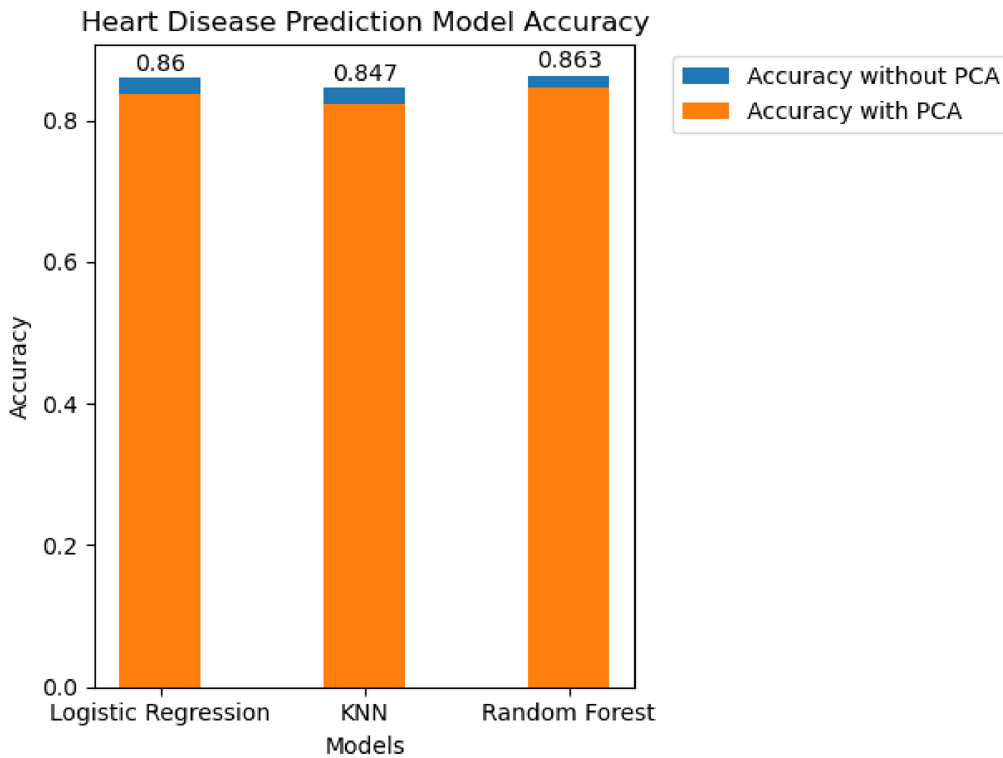



Fig: Accuracy measure of the Models

# CONCLUSION

The paper presents the working of three ML models which are Logistic Regression, KNN algorithm and Random Forest classifier. These algorithms classify the data record of the patient such that they may have a heart stroke or not. The accuracy was found to be highest for the Random Forest model which is an ensemble(hybrid) model. With the growing applications of data science research in medical field, this is one step further in improving medical care for all using modern computer architecture as well as new data analysis techniques. In the future this accuracy can further be improved by using more accurate ML as well as DL classifiers. Further, usage of image dataset to train our model can further increase the real time applications. Further as a future scope, we can find the features with highest weightage in influencing the result for a particular sample and suggest a proper medication based on it. Thus, we were able to train ML models that are able to make early heart disease prediction which is very helpful in saving numerous lives which are risked by this disease by providing early treatment to those diagonised with the complication.

# REFERENCES


[1] M. J. A. Junaid and R. Kumar, "Data Science And Its Application In Heart Disease Prediction," 2020 International Conference on Intelligent Engineering and Management (ICIEM), London, UK, 2020, pp. 396-400, doi: 10.1109/ICIEM48762.2020.9160056.

[2] K. Yuan, L. Yang, Y. Huang and Z. Li, "Heart Disease Prediction Algorithm Based on Ensemble Learning," 2020 7th International Conference on Dependable Systems and Their Applications (DSA), Xi'an, China, 2020, pp. 293-298, doi: 10.1109/DSA51864.2020.00052.

[3] Gavhane, A., Kokkula, G., Pandya, I., & Devadkar, P. K. (2018). Prediction of Heart Disease Using Machine Learning. 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA). doi:10.1109/iceca.2018.8474922

[4] Kavitha, M., Gnaneswar, G., Dinesh, R., Sai, Y. R., & Suraj, R. S. (2021). Heart Disease Prediction using Hybrid machine Learning Model. 2021 6th International Conference on Inventive Computation Technologies (ICICT). doi:10.1109/icict50816.2021.9358597

[5] Purushottam, Saxena, K., & Sharma, R. (2015). Efficient heart disease prediction system using decision tree. International Conference on Computing, Communication & Automation. doi:10.1109/ccaa.2015.7148346

[6] Liu, Y., Zhang, M., Fan, Z., & Chen, Y. (2020). Heart disease prediction based on random forest and LSTM. 2020 2nd International Conference on Information Technology and Computer Application (ITCA). doi:10.1109/itca52113.2020.00137

[7] V. Sharma, A. Rasool and G. Hajela, "Prediction of Heart disease using DNN," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020, pp. 554-562, doi: 10.1109/ICIRCA48905.2020.9182991.

[8] Chakarverti, M., Yadav, S., & Rajan, R. (2019). Classification Technique for Heart Disease Prediction in Data Mining. 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT). doi:10.1109/icicict46008.2019.8993191

[9] Djerioui, M., Brik, Y., Ladjal, M., & Attallah, B. (2020). Heart Disease prediction using MLP and LSTM models. 2020 International Conference on Electrical Engineering (ICEE). doi:10.1109/icee49691.2020.9249935

# APPENDIX

## Making the necessary library imports

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.decomposition import PCA

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
```

## Dataset Preprocessing

In [2]:
```
df = pd.read_csv("heart_disease.csv")
df.head()
```

Out[2]:

| | Gender | age | education | currentSmoker | cigsPerDay | BPMeds | prevalentStroke | prevalentHyp | diabetes | totChol | sysBP | diaBP | BMI | heartRate |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Male | 39 | postgraduate | 0 | 0.0 | 0.0 | no | 0 | 0 | 195.0 | 106.0 | 70.0 | 26.97 | 80.0 |
| **1** | Female | 46 | primaryschool | 0 | 0.0 | 0.0 | no | 0 | 0 | 250.0 | 121.0 | 81.0 | 28.73 | 95.0 |
| **2** | Male | 48 | uneducated | 1 | 20.0 | 0.0 | no | 0 | 0 | 245.0 | 127.5 | 80.0 | 25.34 | 75.0 |
| **3** | Female | 61 | graduate | 1 | 30.0 | 0.0 | no | 1 | 0 | 225.0 | 150.0 | 95.0 | 28.58 | 65.0 |
| **4** | Female | 46 | graduate | 1 | 23.0 | 0.0 | no | 0 | 0 | 285.0 | 130.0 | 84.0 | 23.10 | 85.0 |

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4238 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Gender           4238 non-null   object 
 1   age              4238 non-null   int64  
 2   education        4133 non-null   object 
 3   currentSmoker    4238 non-null   int64  
 4   cigsPerDay       4209 non-null   float64
 5   BPMeds           4185 non-null   float64
 6   prevalentStroke  4238 non-null   object 
 7   prevalentHyp     4238 non-null   int64  
 8   diabetes         4238 non-null   int64  
 9   totChol          4188 non-null   float64
 10  sysBP            4238 non-null   float64
 11  diaBP            4238 non-null   float64
 12  BMI              4219 non-null   float64
 13  heartRate        4237 non-null   float64
 14  glucose          3850 non-null   float64
 15  Heart_ stroke    4238 non-null   object 
dtypes: float64(8), int64(4), object(4)
memory usage: 529.9+ KB
```

In [4]:
```
df.describe()
```

Out[4]:

| | age | currentSmoker | cigsPerDay | BPMeds | prevalentHyp | diabetes | totChol | sysBP | diaBP | BMI | heartR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 4238.000000 | 4238.000000 | 4209.000000 | 4185.000000 | 4238.000000 | 4238.000000 | 4188.000000 | 4238.000000 | 4238.000000 | 4219.000000 | 4237.000 |
| **mean** | 49.584946 | 0.494101 | 9.003089 | 0.029630 | 0.310524 | 0.025720 | 236.721585 | 132.352407 | 82.893464 | 25.802008 | 75.878 |
| **std** | 8.572160 | 0.500024 | 11.920094 | 0.169584 | 0.462763 | 0.158316 | 44.590334 | 22.038097 | 11.910850 | 4.080111 | 12.026 |
| **min** | 32.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 107.000000 | 83.500000 | 48.000000 | 15.540000 | 44.000 |
| **25%** | 42.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 206.000000 | 117.000000 | 75.000000 | 23.070000 | 68.000 |
| **50%** | 49.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 234.000000 | 128.000000 | 82.000000 | 25.400000 | 75.000 |
| **75%** | 56.000000 | 1.000000 | 20.000000 | 0.000000 | 1.000000 | 0.000000 | 263.000000 | 144.000000 | 89.875000 | 28.040000 | 83.000 |
| **max** | 70.000000 | 1.000000 | 70.000000 | 1.000000 | 1.000000 | 1.000000 | 696.000000 | 295.000000 | 142.500000 | 56.800000 | 143.000 |

In [5]:

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

Out[5]:

```
Gender              0
age                 0
education         105
currentSmoker       0
cigsPerDay         29
BPMeds             53
prevalentStroke     0
prevalentHyp        0
diabetes            0
totChol            50
sysBP               0
diaBP               0
BMI                19
heartRate           1
glucose           388
Heart_ stroke       0
dtype: int64
```

In [6]:

```
#Fill null values

df['glucose'] = df.glucose.fillna(df.glucose.median())
df['cigsPerDay'] = df.cigsPerDay.fillna(df.cigsPerDay.median())
df['BMI'] = df.BMI.fillna(df.BMI.median())
df['totChol'] = df.totChol.fillna(df.totChol.median())
df['heartRate'] = df.heartRate.fillna(df.heartRate.median())
```

In [7]:

```
df['education'] = df.education.fillna(df.education.mode().iloc[0])
df['BPMeds'] = df.BPMeds = df.BPMeds.fillna(df.BPMeds.mode().iloc[0])
```

In [8]:

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

Out[8]:

```
Gender             0
age                0
education          0
currentSmoker      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
heartRate          0
glucose            0
Heart_ stroke      0
dtype: int64
```

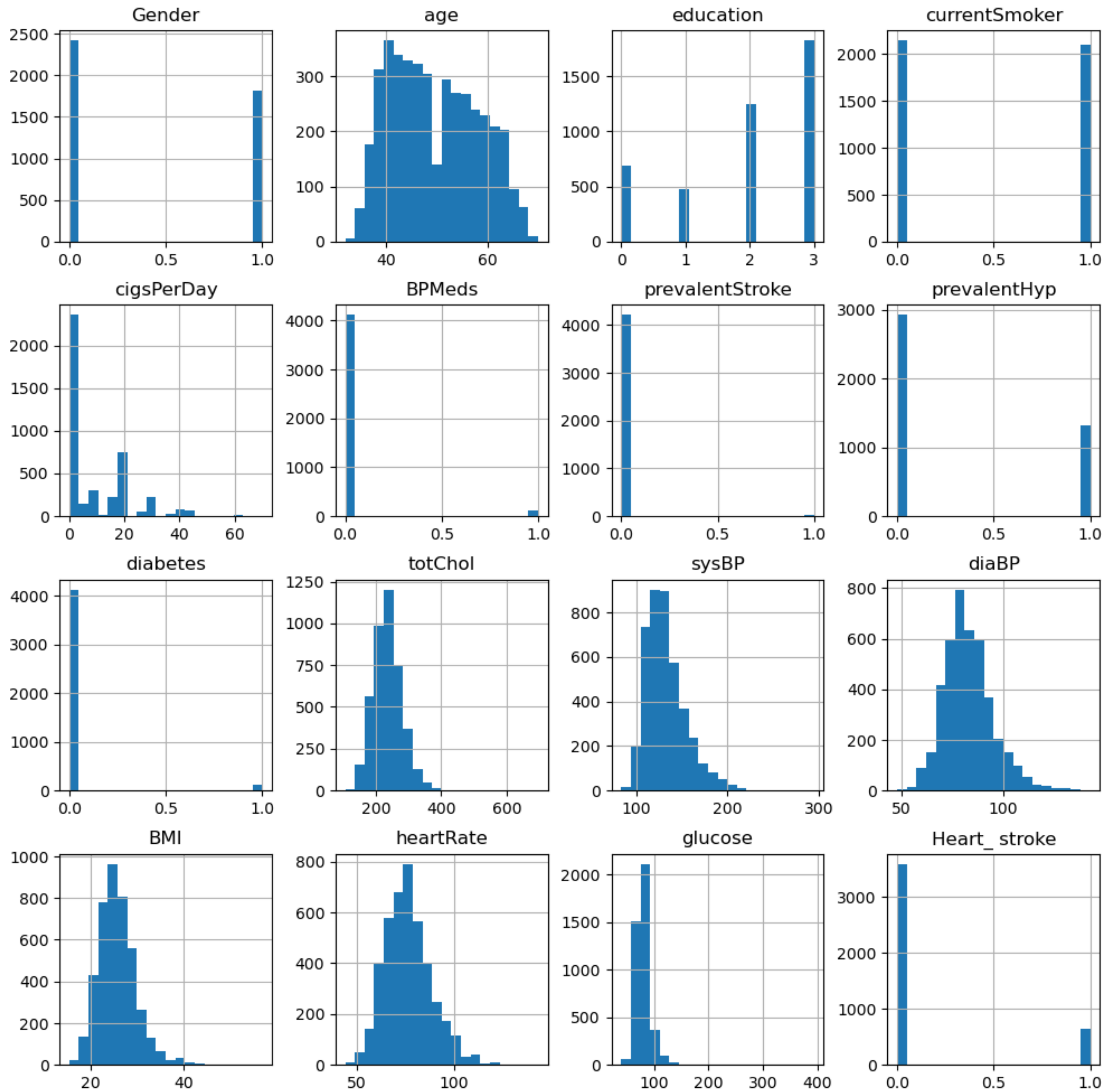
In [9]:

```
le = LabelEncoder()

df['Gender'] = le.fit_transform(df['Gender'])
df['education'] = le.fit_transform(df['education'])
df['prevalentStroke'] = le.fit_transform(df['prevalentStroke'])
df['Heart_ stroke'] = le.fit_transform(df['Heart_ stroke'])
```

# Exploratory Data Analysis (EDA)

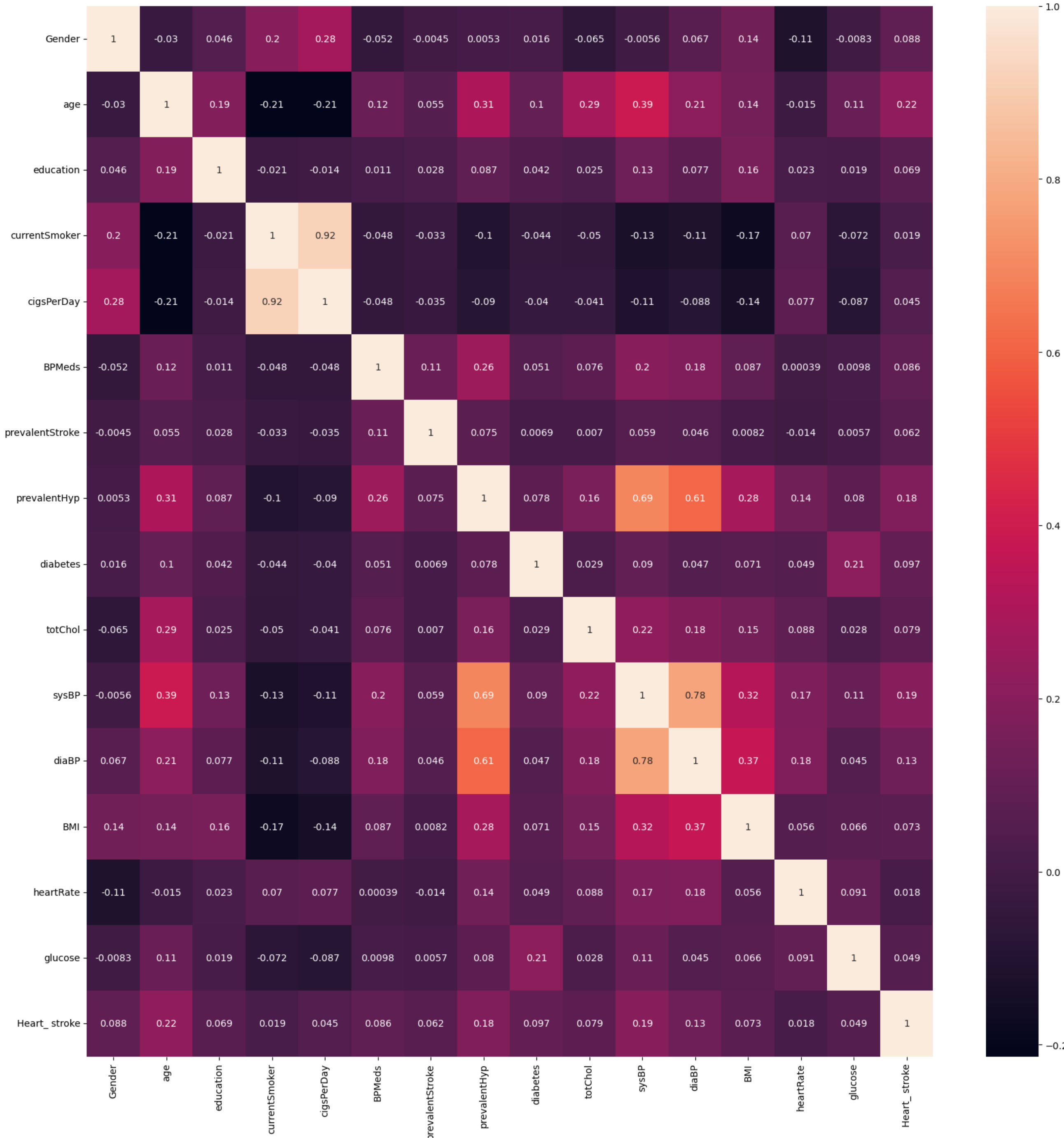
In [10]:

```
df.hist(bins=20, figsize=(12,12))
plt.show()
```




In [11]:
```
fig,ax = plt.subplots(figsize = (20,20))
sns.heatmap(df.corr(method = 'spearman'), annot = True)
```

Out[11]: <AxesSubplot:>




# Dataset Split into Training and Testing

In [10]:
```
df.head()
```

Out[10]:

| | Gender | age | education | currentSmoker | cigsPerDay | BPMeds | prevalentStroke | prevalentHyp | diabetes | totChol | sysBP | diaBP | BMI | heartRate | glu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 39 | 1 | 0 | 0.0 | 0.0 | 0 | 0 | 0 | 195.0 | 106.0 | 70.0 | 26.97 | 80.0 | |
| **1** | 0 | 46 | 2 | 0 | 0.0 | 0.0 | 0 | 0 | 0 | 250.0 | 121.0 | 81.0 | 28.73 | 95.0 | |
| **2** | 1 | 48 | 3 | 1 | 20.0 | 0.0 | 0 | 0 | 0 | 245.0 | 127.5 | 80.0 | 25.34 | 75.0 | |
| **3** | 0 | 61 | 0 | 1 | 30.0 | 0.0 | 0 | 1 | 0 | 225.0 | 150.0 | 95.0 | 28.58 | 65.0 | |
| **4** | 0 | 46 | 0 | 1 | 23.0 | 0.0 | 0 | 0 | 0 | 285.0 | 130.0 | 84.0 | 23.10 | 85.0 | |

In [11]:
```
corr_matrix = df.corr()
corr_matrix['Heart_ stroke'].sort_values(ascending = False)
```

Out[11]:
```
Heart_ stroke      1.000000
age                0.225256
sysBP              0.216429
prevalentHyp       0.177603
diaBP              0.145299
glucose            0.121277
diabetes           0.097317
Gender             0.088428
BPMeds             0.086417
totChol            0.081566
BMI                0.074217
prevalentStroke    0.061810
cigsPerDay         0.058859
education          0.058036
heartRate          0.022857
currentSmoker      0.019456
Name: Heart_ stroke, dtype: float64
```

In [12]:
```
X = df.drop(['Heart_ stroke', 'currentSmoker', 'heartRate', 'education'
            ,'cigsPerDay', 'prevalentStroke'],axis=1)
y = df['Heart_ stroke']
```

In [13]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [34]:
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size = .3, random_state = 68)
```

In [15]:
```
pca = PCA(.95)
X_pca = pca.fit_transform(X)
X_pca.shape
```

Out[15]:
```
(4238, 3)
```

In [49]:
```
X_train_pca, X_test_pca, y_train_pca, y_test_pca = train_test_split(X_pca,y, test_size=.3, random_state = 5)
```

In [17]:
```
print("Without PCA, X_train: ", X_train.shape)
print("With PCA, X_train: ", X_train_pca.shape)
```

```
Without PCA, X_train:  (2966, 10)
With PCA, X_train:  (2966, 3)
```

# Logistic Regression

In [35]:
```
LogReg = LogisticRegression(solver='liblinear')
LogReg_pca = LogisticRegression(solver = 'liblinear')
```

In [36]:
```
LogReg.fit(X_train,y_train)
```

Out[36]:
```
LogisticRegression(solver='liblinear')
```

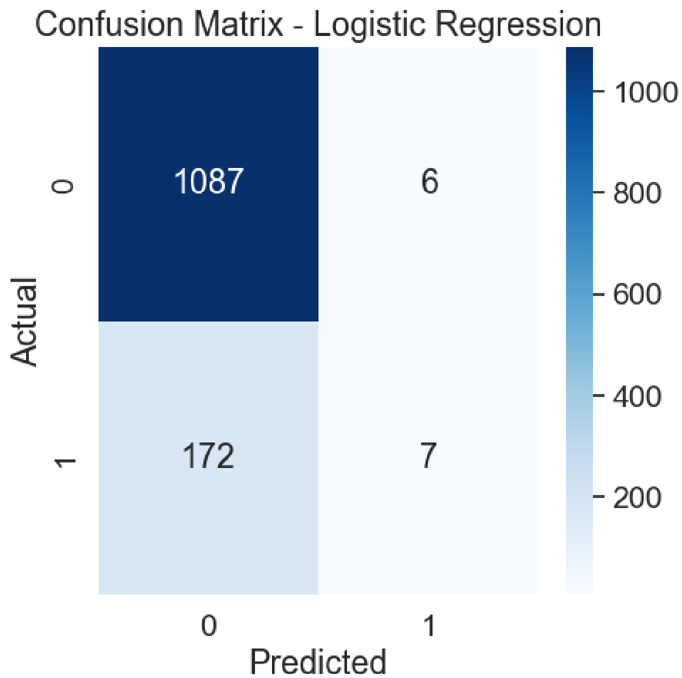
In [50]:
```
LogReg_pca.fit(X_train_pca, y_train_pca)
```

Out[50]:
```
LogisticRegression(solver='liblinear')
```

In [37]:
```
accLogReg = LogReg.score(X_test, y_test)
print(accLogReg)
```

```
0.860062893081761
```

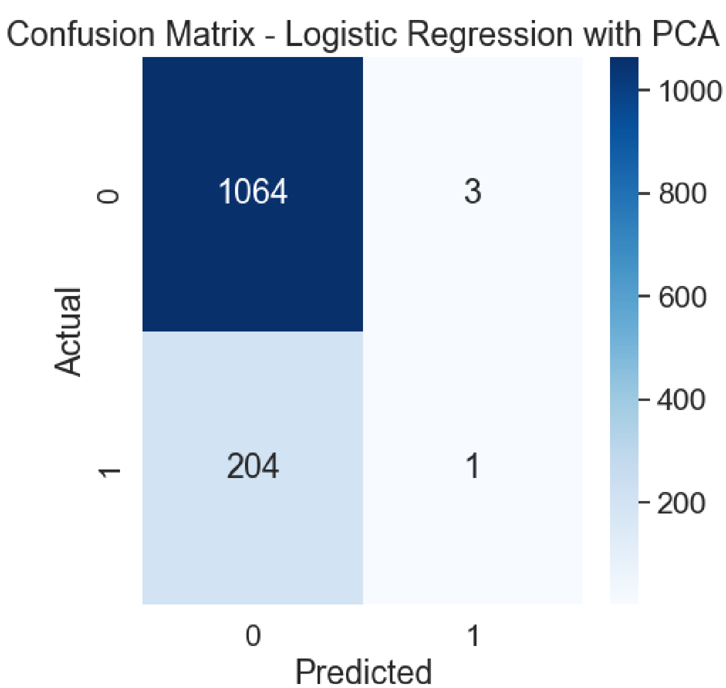
In [59]:
```
y_pred = LogReg.predict(X_test)

plt.figure(figsize=(5,5))
sns.set(font_scale=1.4)
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Logistic Regression')
plt.show()
```




In [51]:
```
accLogRegwPCA = LogReg_pca.score(X_test_pca, y_test_pca)
print(accLogRegwPCA)
```

```
0.8372641509433962
```

In [58]:
```
y_pred_pca = LogReg_pca.predict(X_test_pca)

plt.figure(figsize=(5,5))
sns.set(font_scale=1.4)
sns.heatmap(confusion_matrix(y_test_pca, y_pred_pca), annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Logistic Regression with PCA')
plt.show()
```




## K Nearest Neighbor

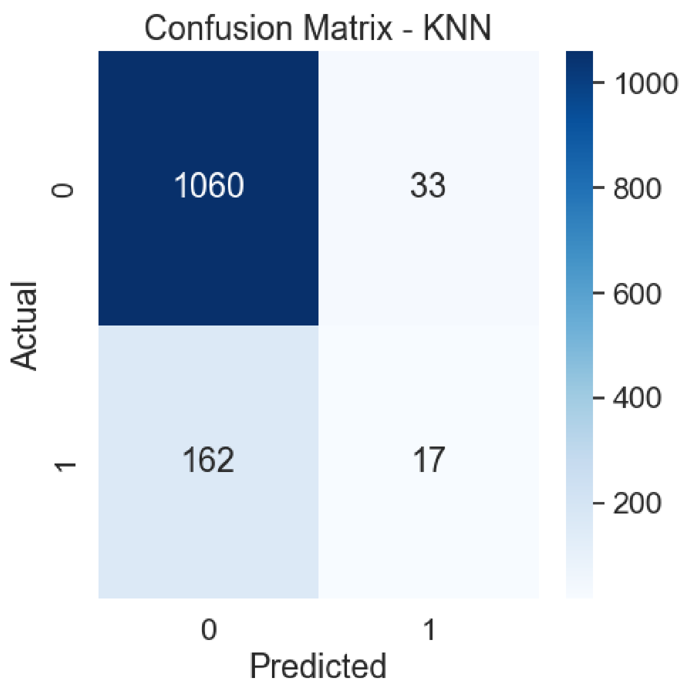
In [38]:
```
knn = KNeighborsClassifier(n_neighbors = 5)
knn_pca = KNeighborsClassifier(n_neighbors = 5)
```

In [39]:
```
knn.fit(X_train,y_train)
```
Out[39]:
```
KNeighborsClassifier()
```

In [52]:
```
knn_pca.fit(X_train_pca, y_train_pca)
```
Out[52]:
```
KNeighborsClassifier()
```

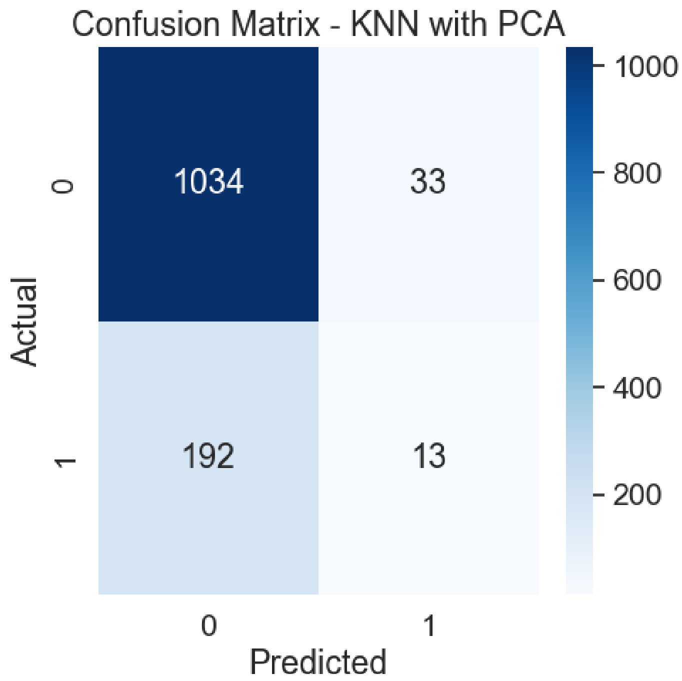
In [41]:
```
accKNN = knn.score(X_test, y_test)
print(accKNN)
```
```
0.8466981132075472
```

In [31]:
```
y_pred = knn.predict(X_test)

plt.figure(figsize=(5,5))
sns.set(font_scale=1.4)
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - KNN')
plt.show()
```




In [53]:
```
accKNNwPCA = knn_pca.score(X_test_pca, y_test_pca)
print(accKNNwPCA)
```
```
0.8231132075471698
```

In [33]:
```
y_pred_pca = knn_pca.predict(X_test_pca)

plt.figure(figsize=(5,5))
sns.set(font_scale=1.4)
sns.heatmap(confusion_matrix(y_test_pca, y_pred_pca), annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - KNN with PCA')
plt.show()
```




# Random Forest

In [42]:
```
RF = RandomForestClassifier(n_estimators=100)
RF_pca = RandomForestClassifier(n_estimators=100)
```

In [46]:
```
RF.fit(X_train, y_train)
```

Out[46]:
```
RandomForestClassifier()
```

In [54]:
```
RF_pca.fit(X_train_pca, y_train_pca)
```

Out[54]:
```
RandomForestClassifier()
```

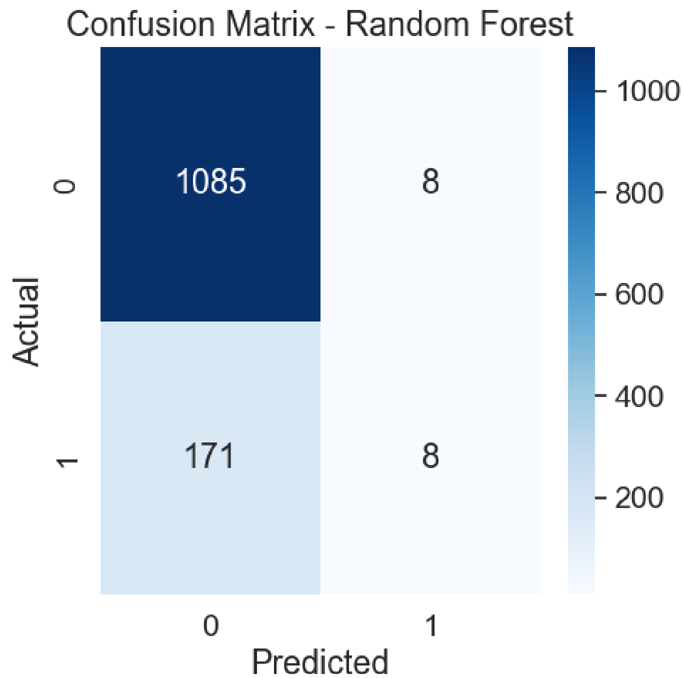
In [47]:
```
accRF = RF.score(X_test, y_test)
print(accRF)
```

```
0.8632075471698113
```

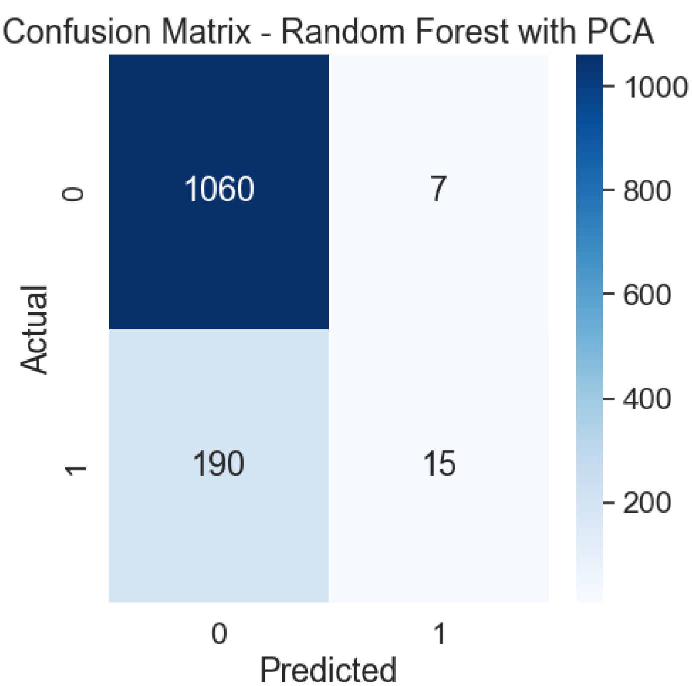
In [55]:
```
accRFwPCA = RF_pca.score(X_test_pca, y_test_pca)
print(accRFwPCA)
```

```
0.845125786163522
```

In [39]:
```
y_pred = RF.predict(X_test)

plt.figure(figsize=(5,5))
sns.set(font_scale=1.4)
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Random Forest')
plt.show()
```



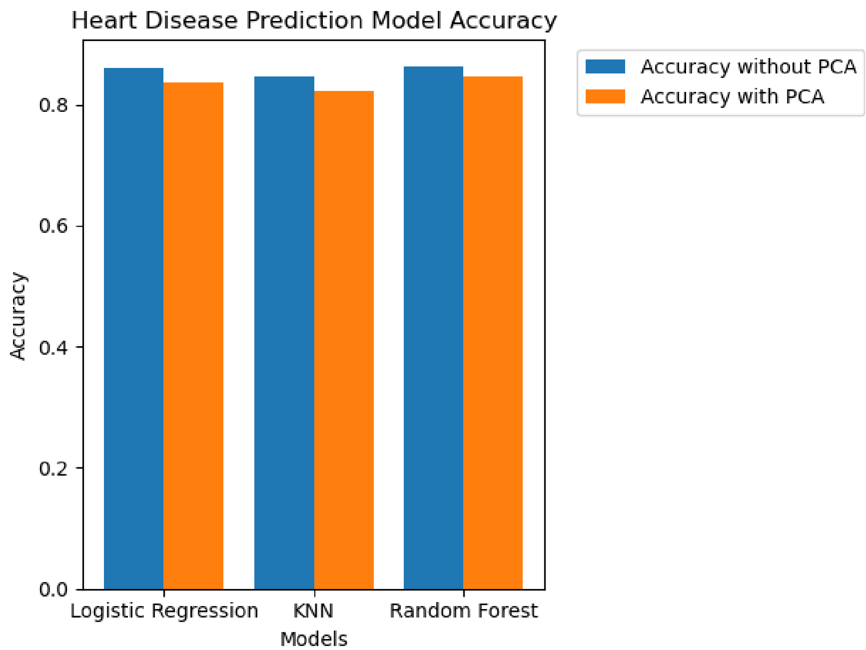

In [41]:
```
y_pred_pca = RF_pca.predict(X_test_pca)

plt.figure(figsize=(5,5))
sns.set(font_scale=1.4)
sns.heatmap(confusion_matrix(y_test_pca, y_pred_pca), annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Random Forest with PCA')
plt.show()
```




## Result Visualization

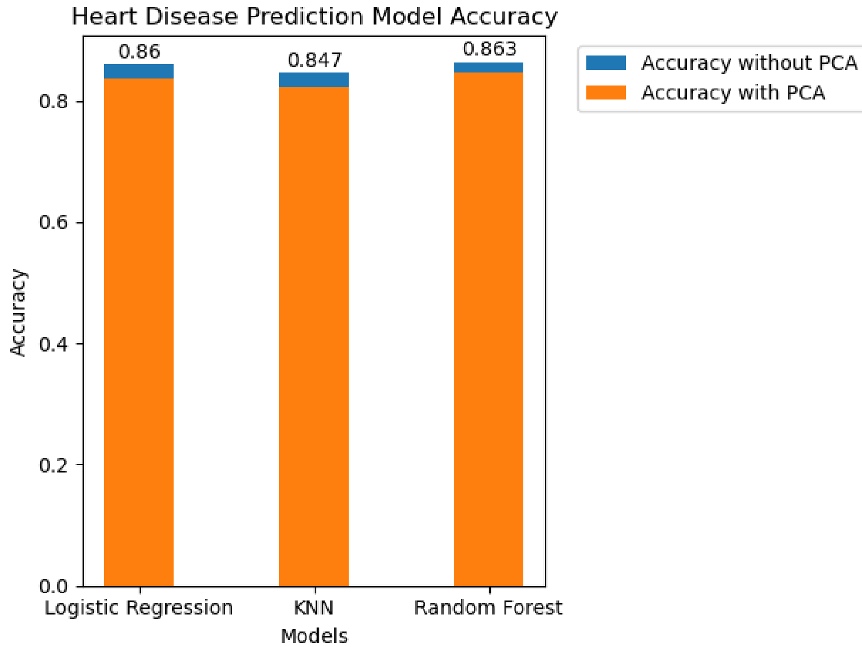
In [56]:
```
models = ['Logistic Regression','KNN','Random Forest']
acc = [accLogReg, accKNN, accRF]
accwPCA = [accLogRegwPCA, accKNNwPCA, accRFwPCA]

X_axis = np.arange(len(models))

plt.bar(X_axis - 0.2, acc, 0.4, label = 'Accuracy without PCA')
```

```
plt.bar(X_axis + 0.2, accwPCA, 0.4, label = 'Accuracy with PCA')

plt.xticks(X_axis, models)
plt.xlabel("Models")
plt.ylabel("Accuracy")
plt.title("Heart Disease Prediction Model Accuracy")
plt.legend()
plt.legend(bbox_to_anchor=(1.05, 1.0), loc='upper left')
plt.tight_layout()
plt.show()
```




In [57]:

```
def addlabels(x,y):
    for i in range(len(x)):
        plt.text(i,y[i]+0.01, round(y[i],3), ha='center')

models = ['Logistic Regression','KNN','Random Forest']
acc = [accLogReg, accKNN, accRF]
accwPCA = [accLogRegwPCA, accKNNwPCA, accRFwPCA]

X_axis = np.arange(len(models))

x=models
y = acc
plt.bar(X_axis, acc, 0.4, label = 'Accuracy without PCA')
plt.bar(X_axis, accwPCA, 0.4, label = 'Accuracy with PCA')

plt.xticks(X_axis, models)
plt.xlabel("Models")
plt.ylabel("Accuracy")
plt.title("Heart Disease Prediction Model Accuracy")
plt.legend()
plt.legend(bbox_to_anchor=(1.05, 1.0), loc='upper left')
plt.tight_layout()
addlabels(x,y)
plt.show()
```


Heart Disease Prediction Model Accuracy
0.86
0.847
0.863
Accuracy without PCA
Accuracy with PCA
0.8
0.6
0.4
0.2
0.0
Accuracy
Logistic Regression
KNN
Random Forest
Models