SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

STREAMLINING ML PIPELINES FOR CLINICAL WORKFLOW IN RISK FACTOR ANALYSIS FOR CARDIOVASCULAR DISEASE (CVD)

INT 300 – INTERNSHIP PROJECT REPORT

Submitted by

JOSELYN DIANA CINDRELLA – E0120017

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence and Machine Learning)

Sri Ramachandra Faculty of Engineering and Technology

Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116

APRIL 2022

STREAMLINING ML PIPELINES FOR CLINICAL WORKFLOW IN RISK FACTOR ANALYSIS FOR CARDIOVASCULAR DISEASE (CVD)

INT 300 – INTERNSHIP PROJECT REPORT

Submitted by

JOSELYN DIANA CINDRELLA – E0120017

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Medical Engineering)

Sri Ramachandra Faculty of Engineering and Technology

Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116

APRIL 2022

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

BONAFIDE CERTIFICATE

Certified that this project report **"Streamlining ML Pipelines for Clinical Workflow in Risk Factor Analysis for Cardiovascular Disease (CVD)"** is the bonafide record of work done by **"Joselyn Diana Cindrella – E0120017"** who carried out the internship work under my supervision.

Signature of the Supervisor

G. Jayanthi Ph. D.,

Assistant Professor,

Department of Computer Science and Engineering

Sri Ramachandra Faculty of Engineering and Technology,

SRIHER, Porur, Chennai-600 116.

Signature of Vice-Principal

Prof. M. Prema

Vice-Principal,

Department of Computer Science and Engineering

Sri Ramachandra Faculty of Engineering and Technology,

SRIHER, Porur, Chennai-600 116.

Evaluation Date:

SRI RAMACHANDRA

INSTITUTE OF HIGHER EDUCATION AND RESEARCH

(Category - I Deemed to be University) Porur, Chennai

SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

ACKNOWLEDGEMENT

I express my sincere gratitude to our Chancellor, Vice-Chancellor and our sincere gratitude to our Provost **Dr. V. Raju** and our Vice-Principal **Prof. M. Prema** for their support and for providing the required facilities for carrying out this study.

I wish to thank my faculty supervisor(s), Prof. **G. Jayanthi** Department of Computer Science and Engineering, Sri Ramachandra faculty of Engineering and Technology for extending help and encouragement throughout the project. Without his/her continuous guidance and persistent help, this project would not have been a success for me.

I am grateful to all the members of Sri Ramachandra Faculty of Engineering and Technology, my beloved parents and friends for extending the support, who helped us to overcome obstacles in the study.

TABLE OF CONTENTS

LIST OF FIGURES

1. ABSTRACT

Cardiovascular disease is a type of disease that affects the heart and blood vessels of people in different age groups. Its risk factors include resting bp, cholesterol, fasting blood sugar, maximum heart rate, average glucose level, smoking status, hyper tension, residence type, work type and so on. To create an awareness among people about the social and biological risk factors that cause cardiovascular disease. We can also use computer aided machines to improve the medical diagnosis. Visualization and machine learning is done for better analysis of the risk factors to create awareness. From the analysis, we get the major risk factors causing the cardiovascular disease like blood pressure, glucose level, work type and smoking status. Future works of this analysis include study of cohort charts using the data collected which has both the details of social and biological scores of the patient.

2. INTRODUCTION

- **Motivation:**
    - To create an awareness among people about the social and biological risk factors that cause cardiovascular disease.
- **Existing Approaches and Need for further study:**
    - The present approaches are done using Deep learning and not machine learning algorithms.
- **Applications & Technologies:**
    - We can use computer aided machines to improve the medical diagnosis which can be developed using the Python programming and we can visualize the data using Tableau and which can also be displayed on storyboard.

3. REVIEW OF LITERATURE / PRODUCT

Author: Muhammad Anwarul Azim, Md Rayhan Kabir, Rasif Ajwad **Title:** Identifying the Risk of Cardiovascular Diseases from the Analysis of Physiological Attributes
Methodology: Analyze the dataset, preprocessed the data using various supervised machine learning algorithms.
Results: Accuracy using KNN and Decision Tree is 86.84 and 78.95 respectively.
Limitation: Need of the usage of deep learning algorithms for better accuracy.
Challenges: Extracting data regarding the ECG patterns and formats.

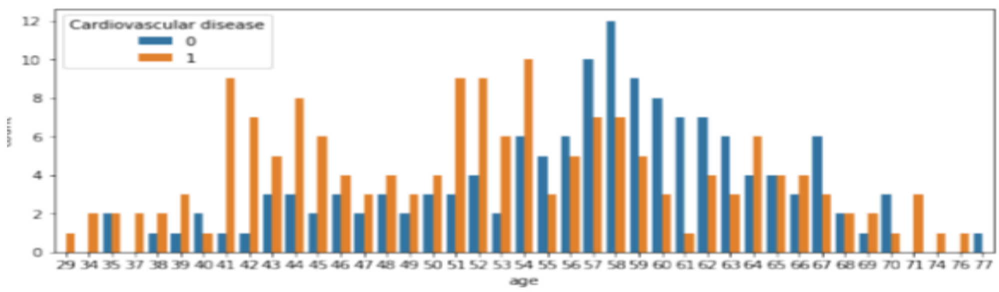


Fig 3.1

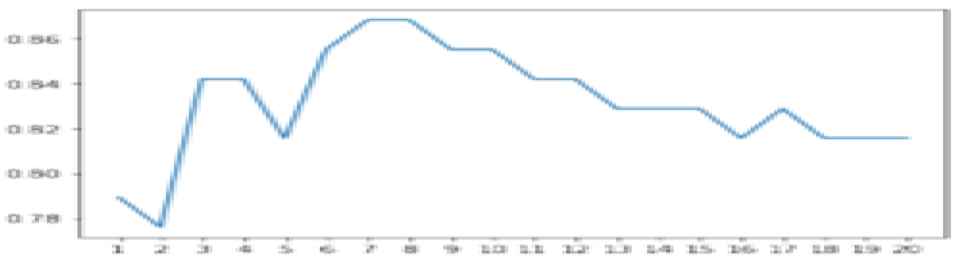


Fig 3.2

4. PROBLEM STATEMENT

Description:

To analyze cardiovascular dataset using Python and Tableau with ML pipeline algorithms to find the risk factors causing heart disease.

OBJECTIVE	METHODOLOGY
1. To collect the data samples	1. Dataset - Kaggle
2. To pre-process the dataset and prepare for ML task	2. Python
3. To visualize the data	3. Tableau
4. To create ML model	4. Python
5. To display all the work	5. Website

4.1 METHODOLOGY 1:

Description:

- Visualizing the data using measures and dimensions using Tableau.
- Creating dash-boards / story-boards.

Workflow diagram:

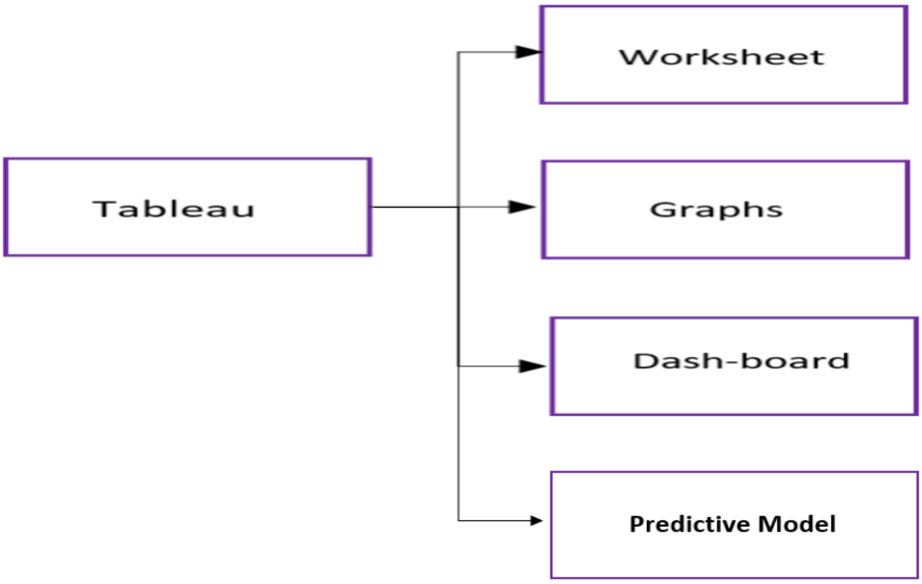


Fig 4.1.1

4.2 METHODOLOGY 2 & 4:

Description:

- Visualization using python.
- Creating ML models to predict accuracy.

Workflow Diagram:

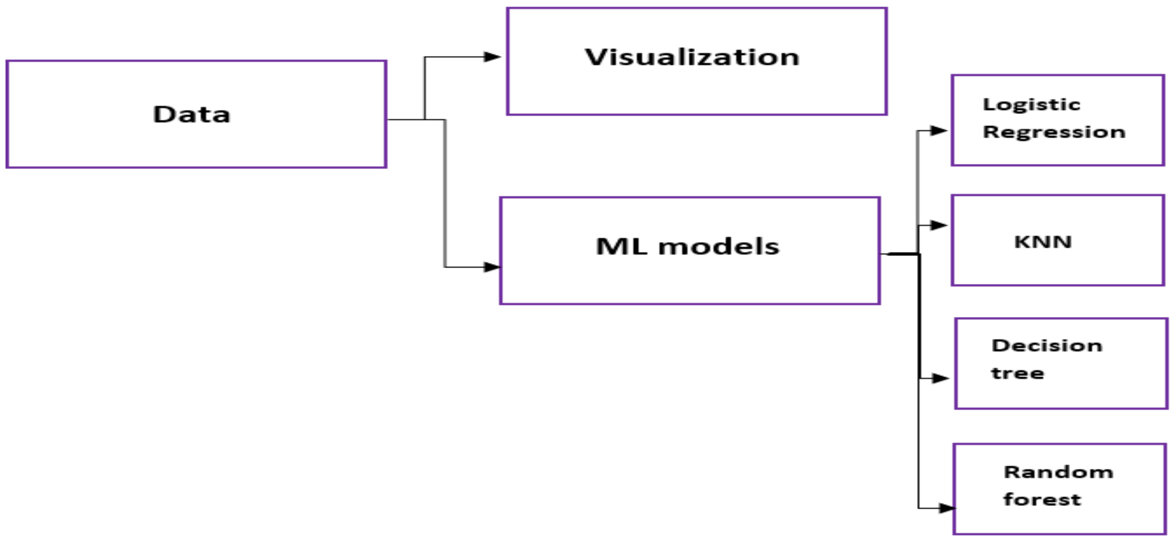


Fig 4.2.1

4.3 METHODOLOGY 3:

Description:

- Visualizing the data using measures and dimensions using Tableau.
- Creating dash-boards / story-boards.

Workflow Diagram:

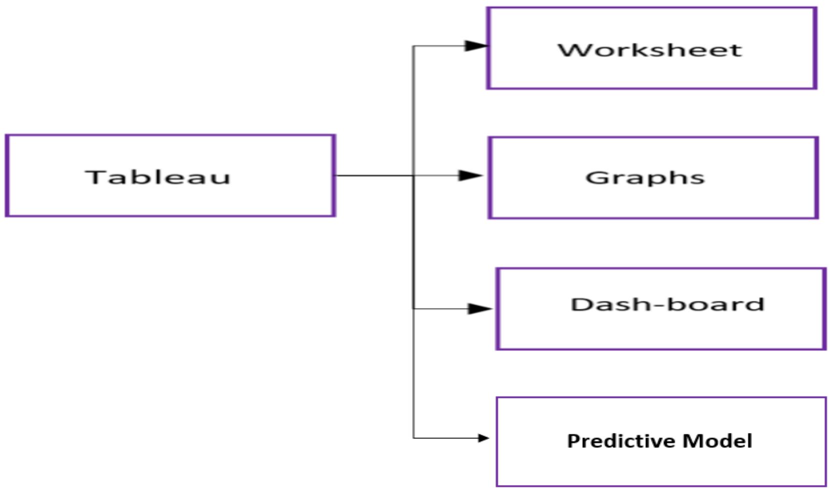


Fig 4.3.1

4.4 METHODOLOGY 5:

Description:

- Webpage.

Workflow Diagram:

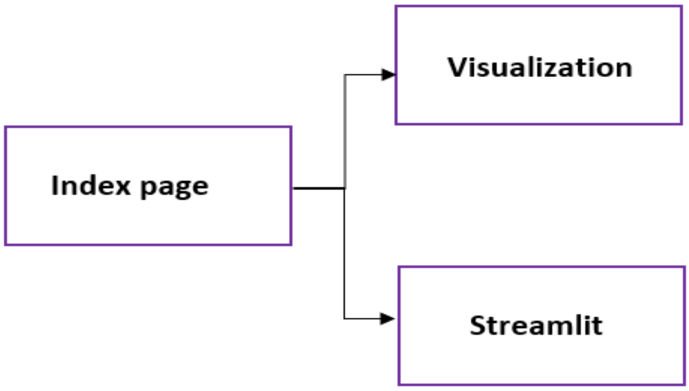


Fig 4.4.1

5. TOOLS AND TECHNOLOGY USED

- **Python:**

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

- **Tableau:**

Tableau is a leading data visualization tool used for data analysis and business intelligence. Gartner's Magic Quadrant classified Tableau as a leader for analytics and business intelligence.

6. Visualization

6.1 Biological Factors

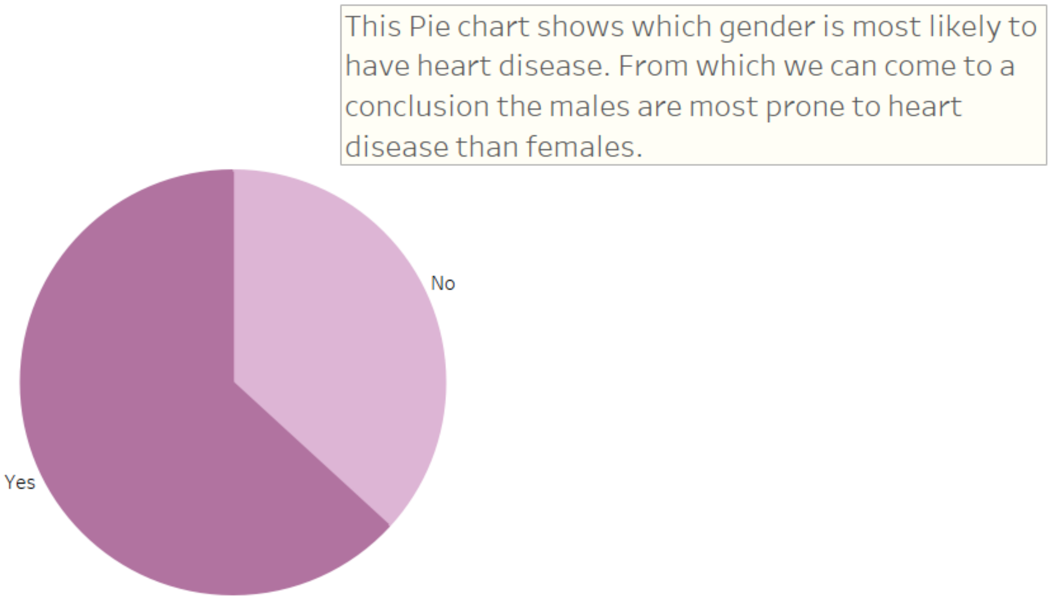


Fig 6.1.1 - Presence of heart disease m vs f

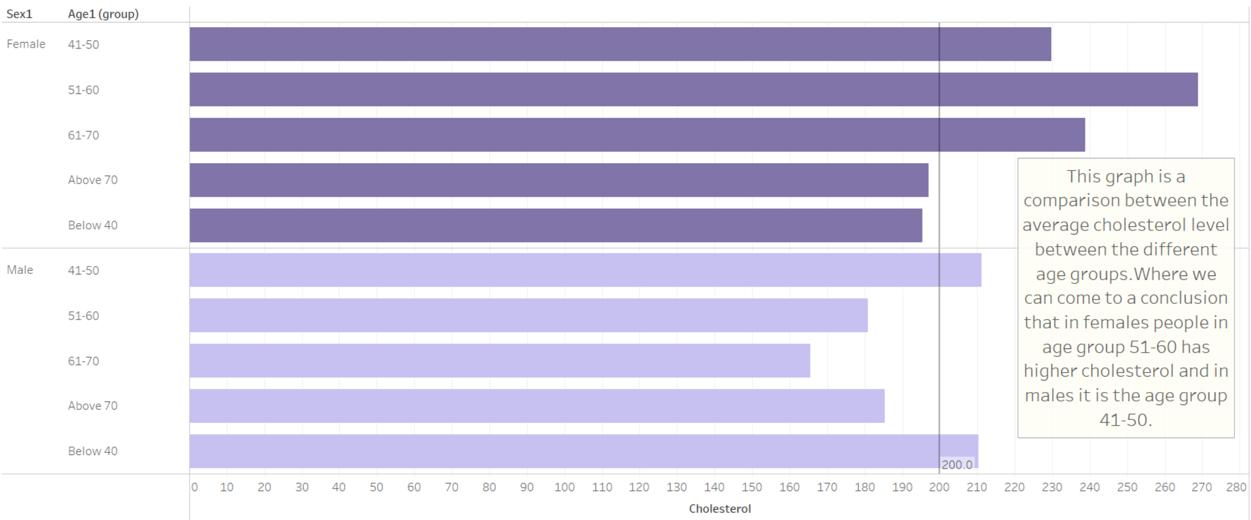


Fig 6.1.2 - Average cholesterol vs diff age grp

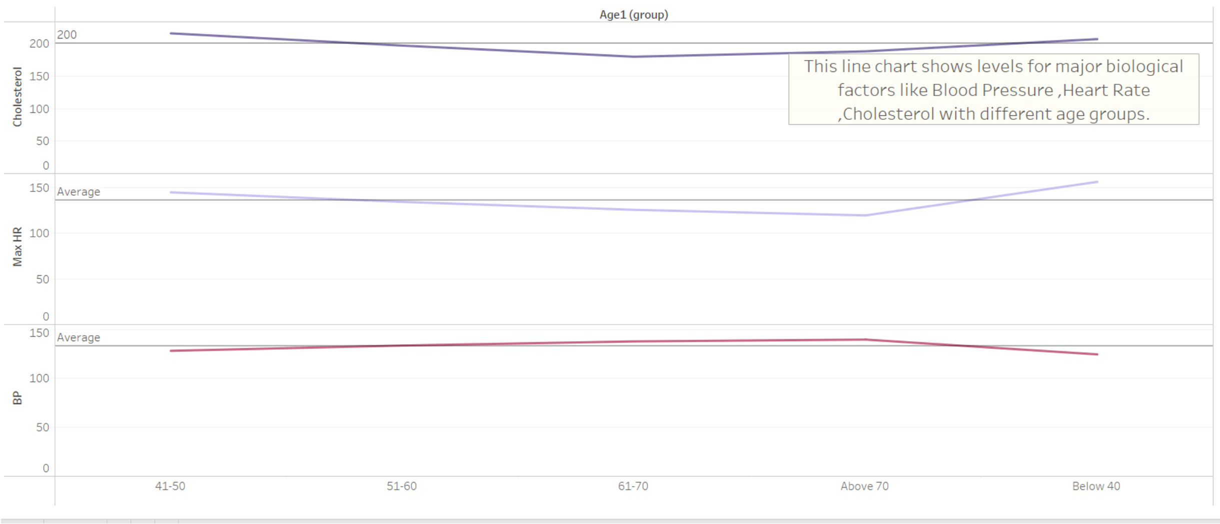


Fig 6.1.3 - Trend of biological factors vs age grp

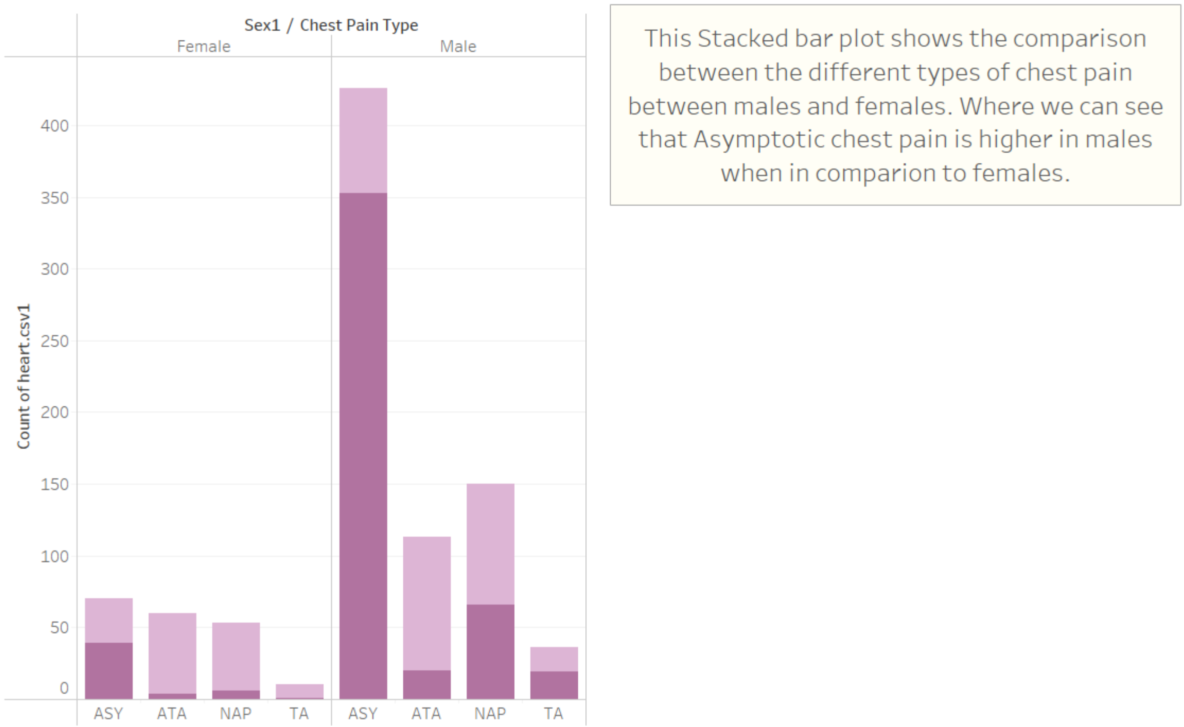


Fig 6.1.4 - Types of chest pain vs no. of m vs f

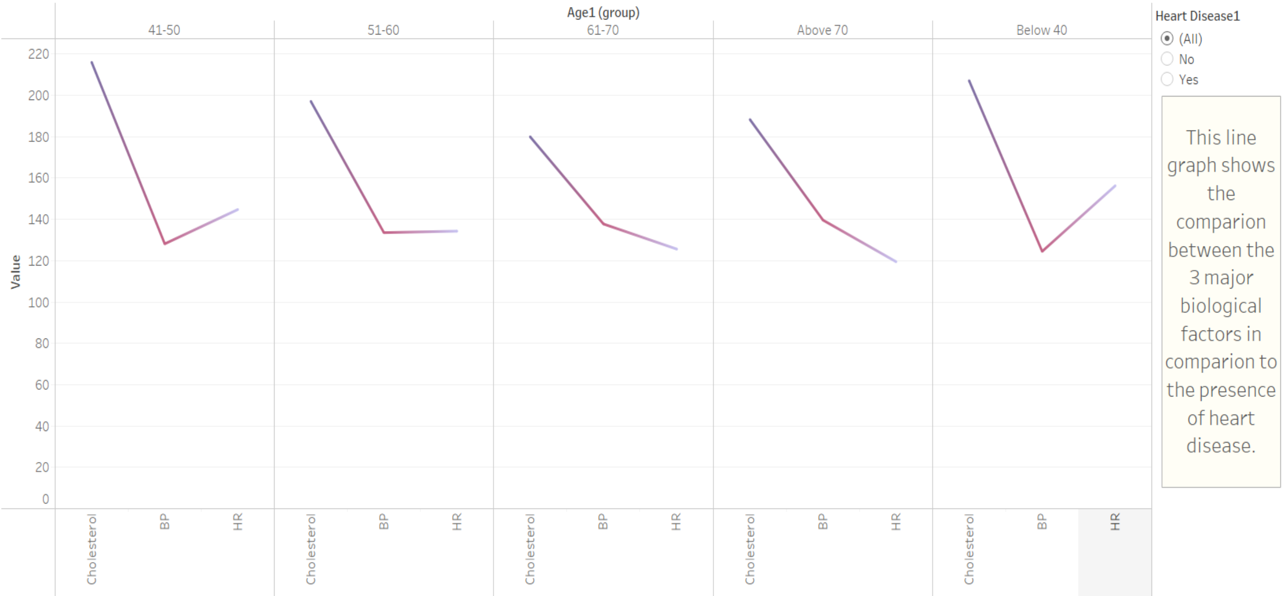


Fig 6.1.5 - Line chart of Chole, BP, HR in diff age grp

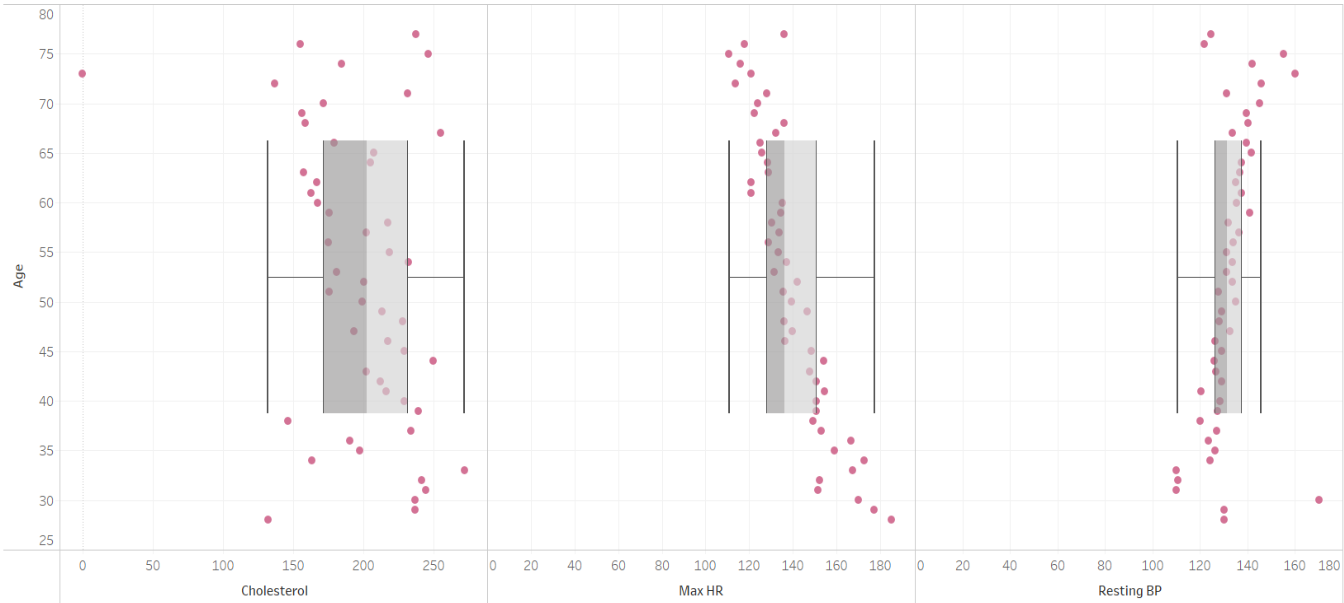


Fig 6.1.6 - Statistics of biological factors using box plot

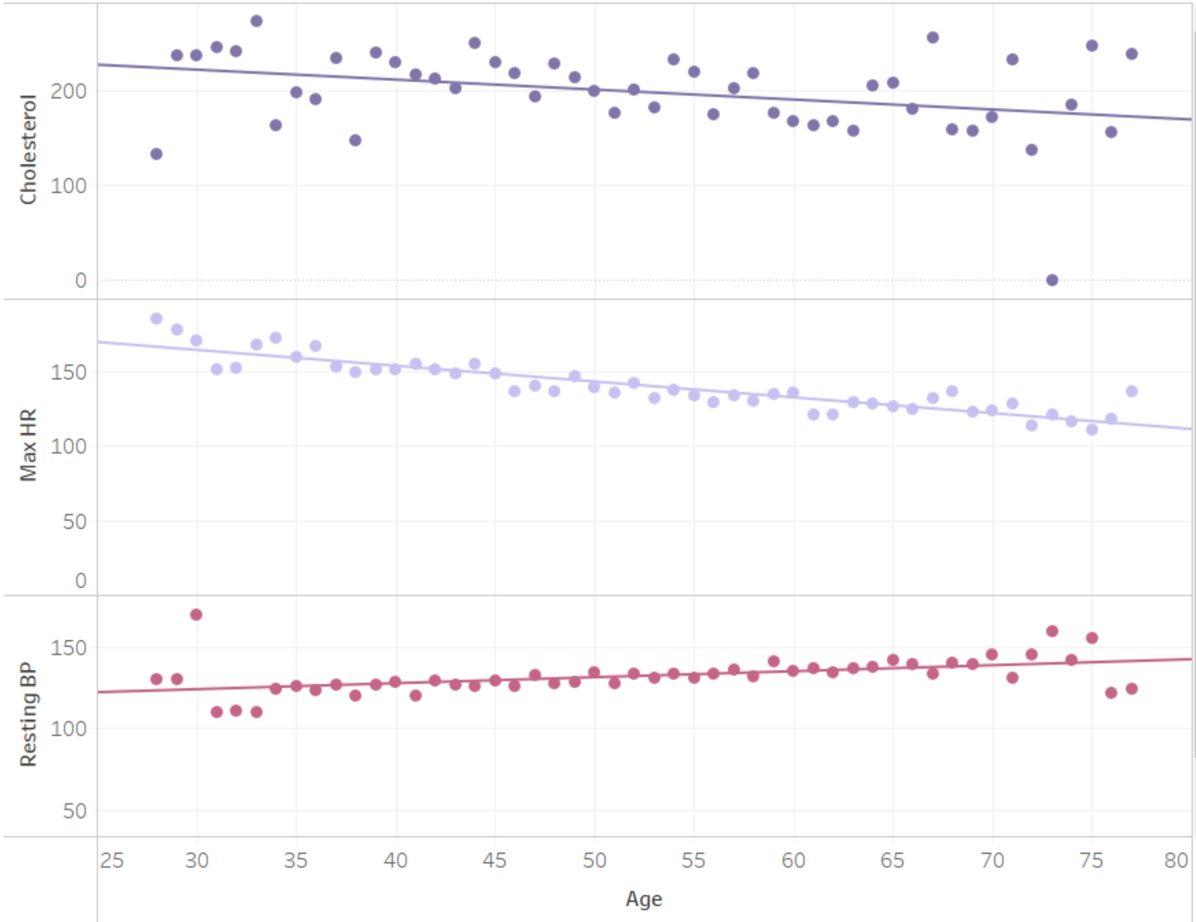


Fig 6.1.7 - Trend line of the same factors

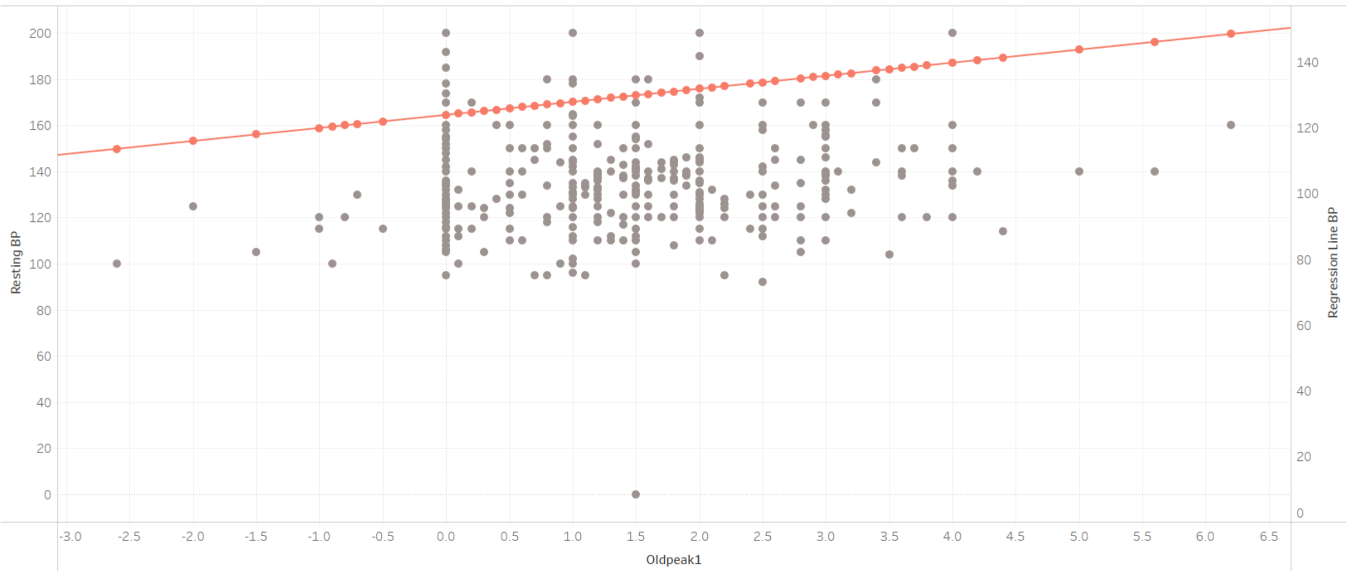


Fig 6.1.8 - Linear Regression for Resting BP

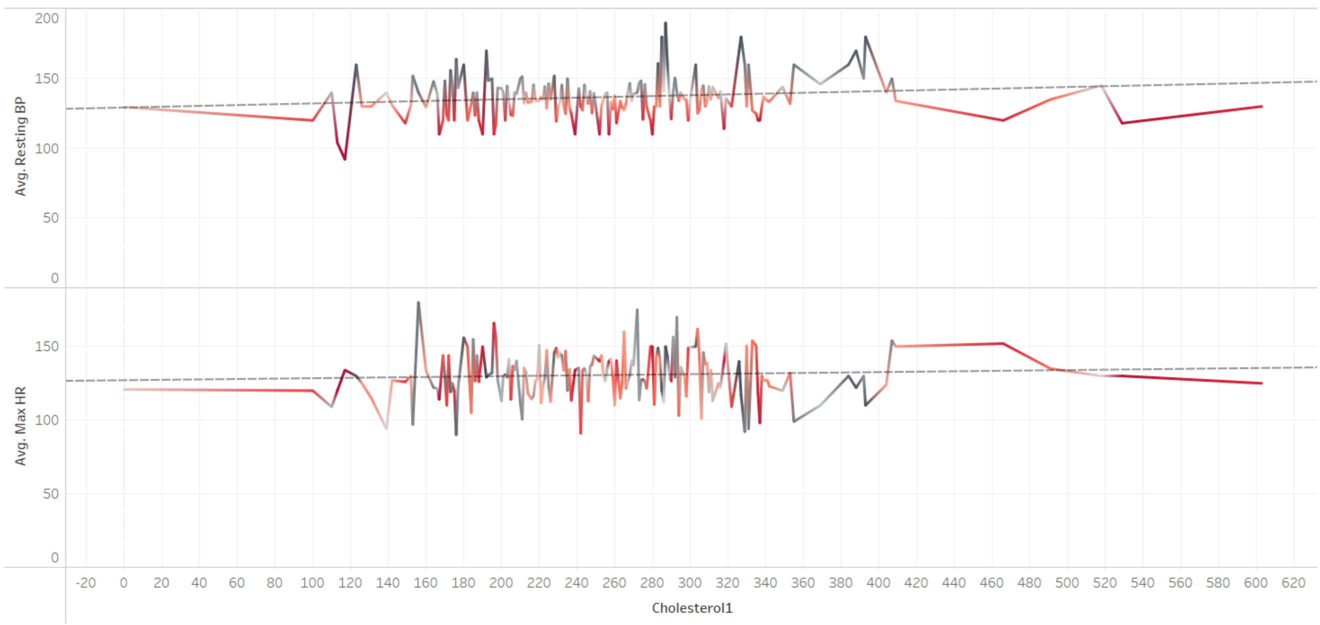


Fig 6.1.9 - Predictive Model

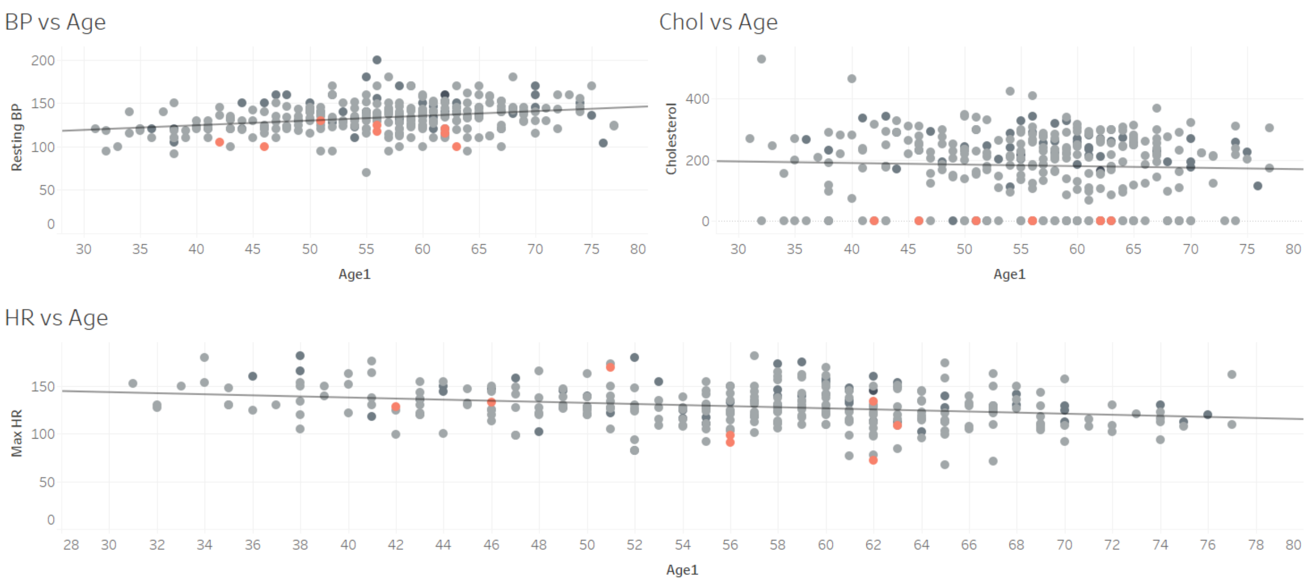


Fig 6.1.10 - Trend line of 3 biological factors with color shading based on old peak

6.2 Social Factors

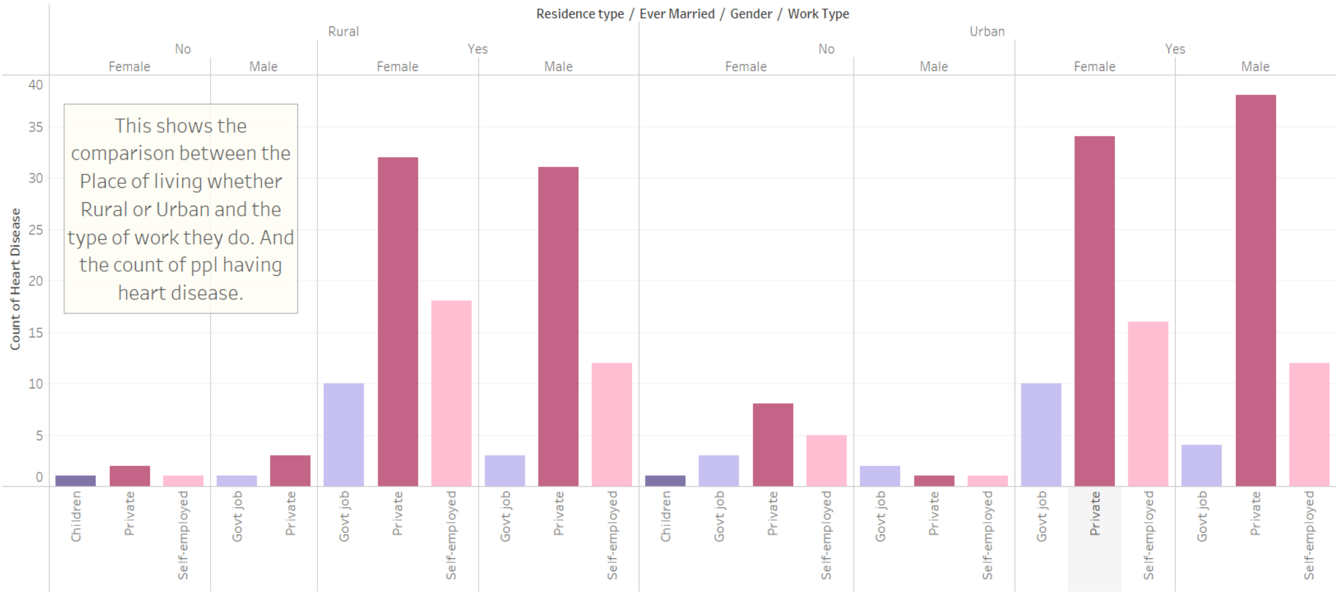


Fig 6.2.1 - Work type vs the no. of people affected

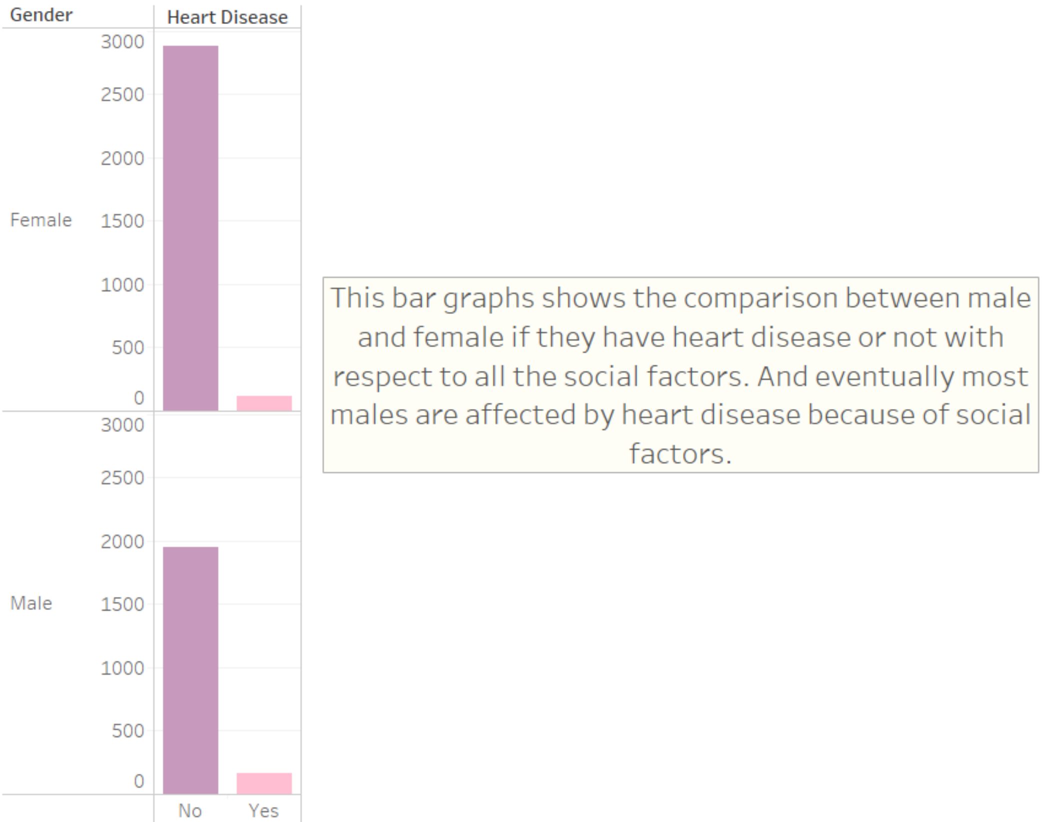


Fig 6.2.2 - All social factors vs affected m and f

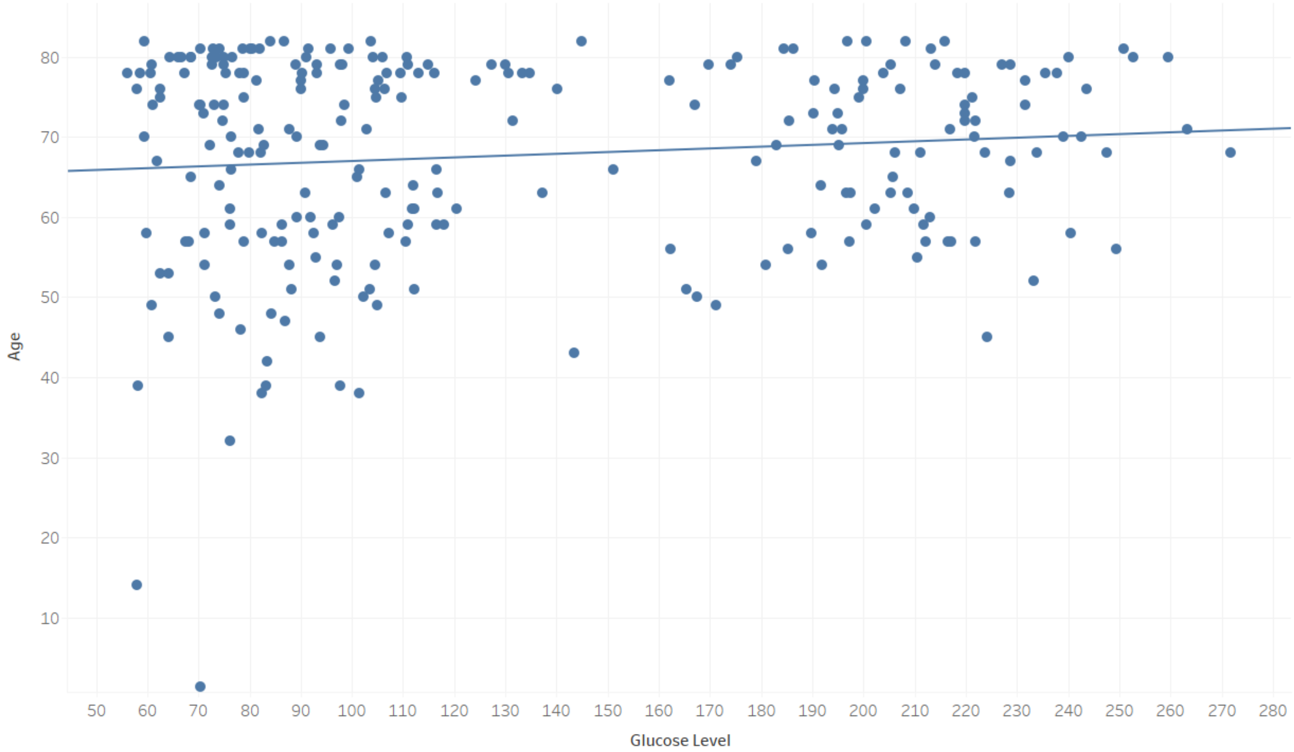


Fig 6.2.3 - Glucose

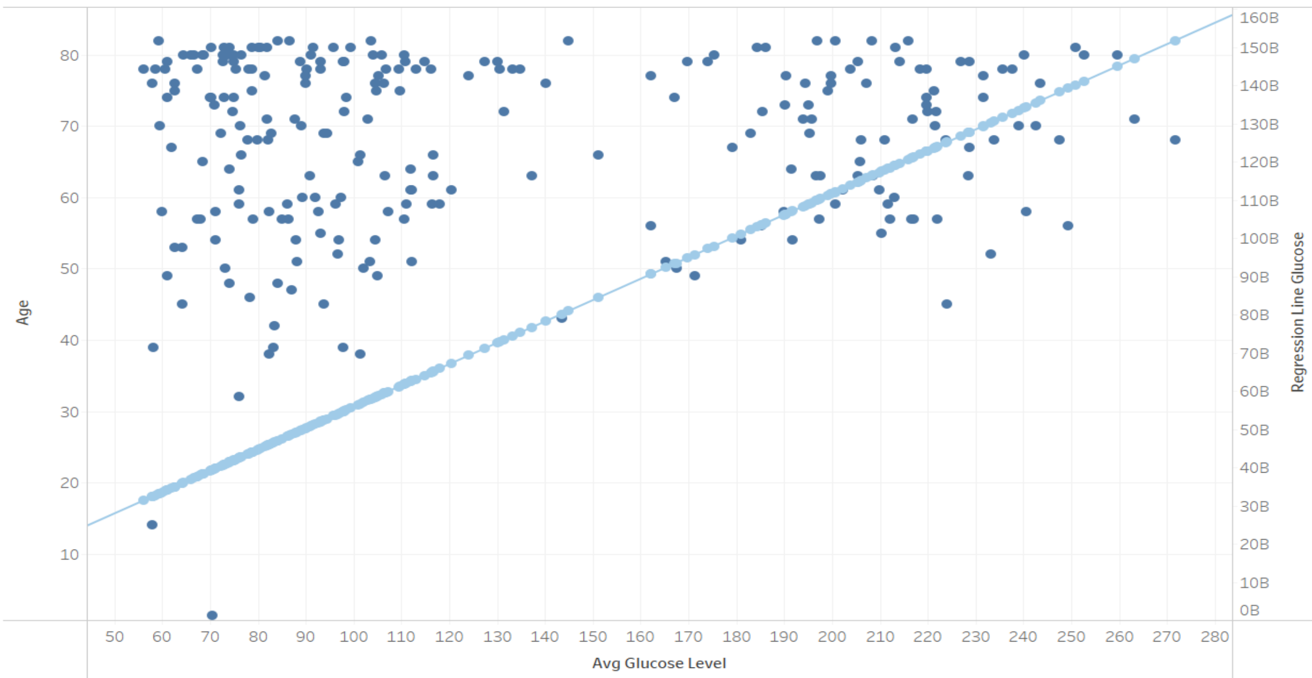


Fig 6.2.4 - Linear Regression-Glucose

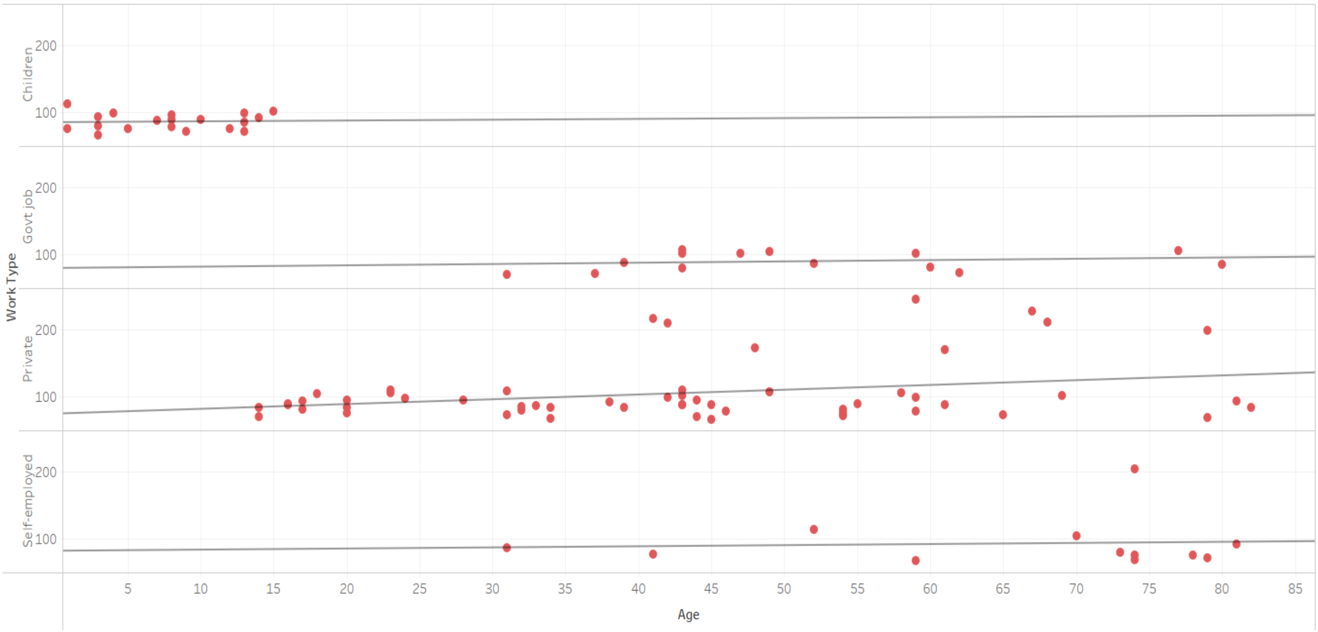


Fig 6.2.5 - Trendline of Age vs Work Type

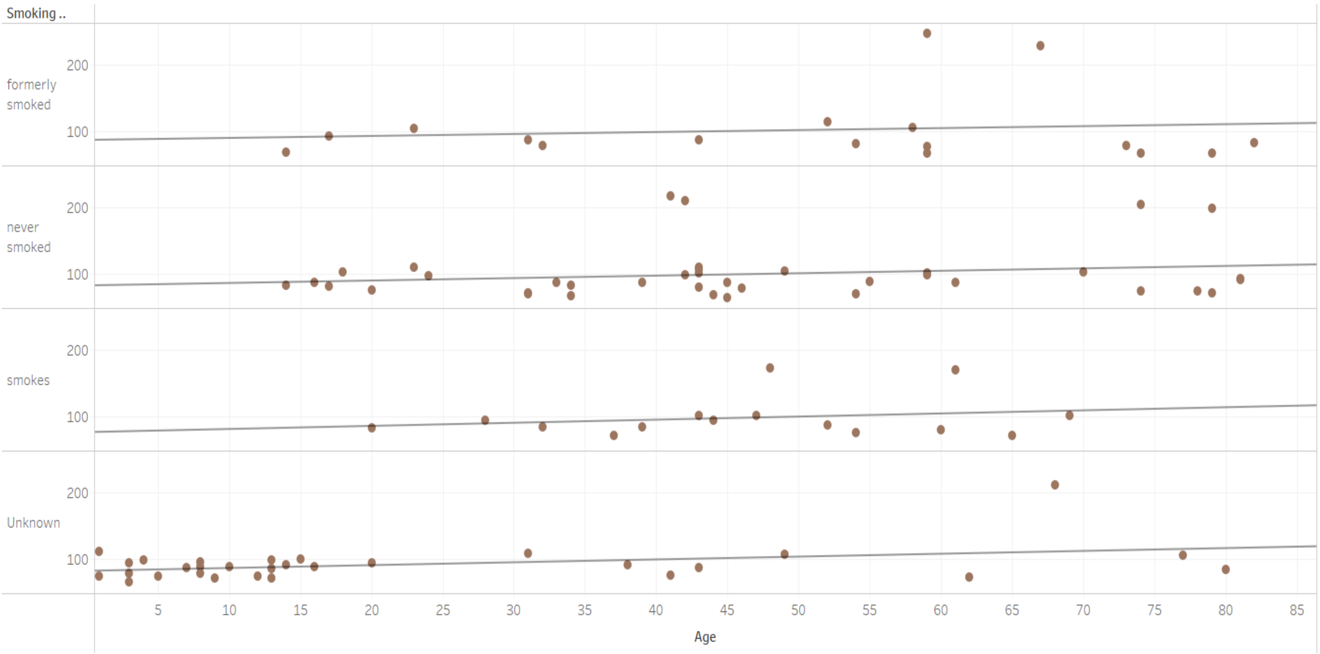


Fig 6.2.6 - Trendline of Age vs Smoking

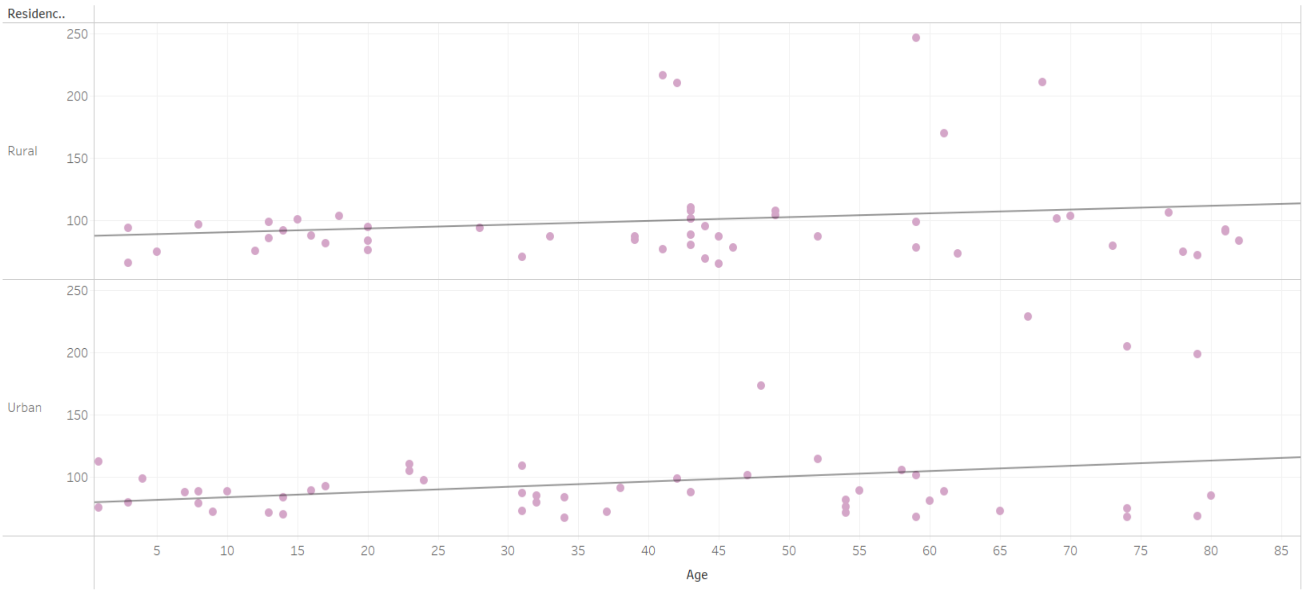


Fig 6.2.7 - Trendline of Age vs Residence

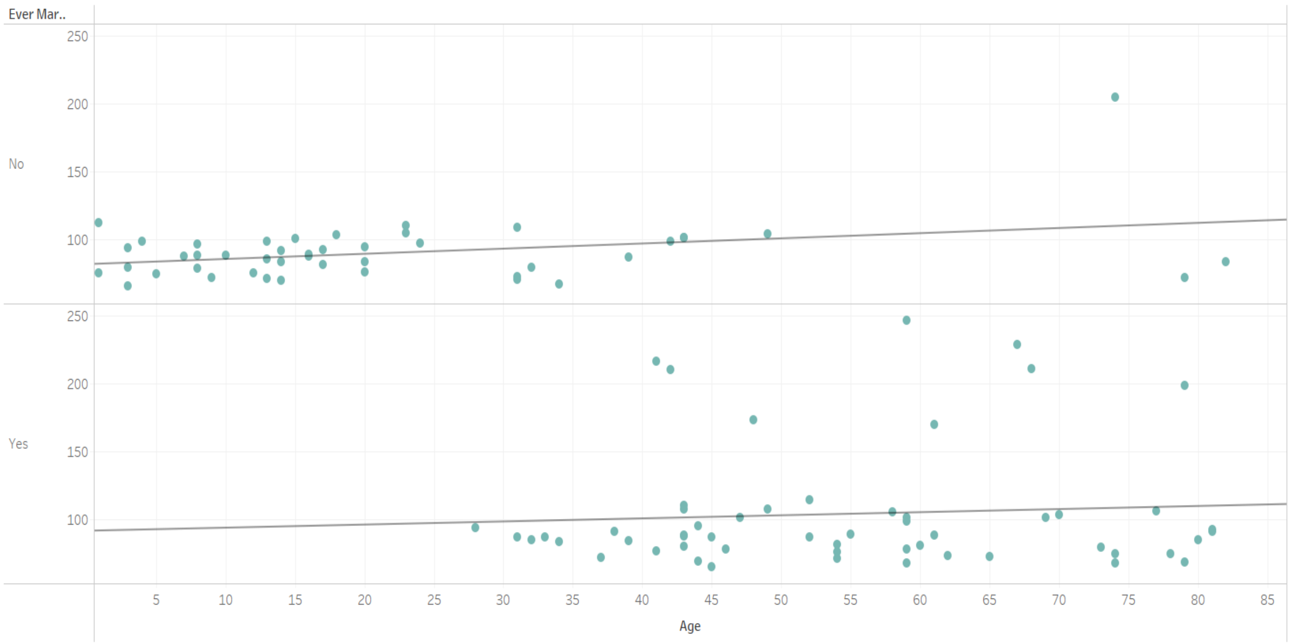


Fig 6.2.8 - Trendline of Age vs Ever Married

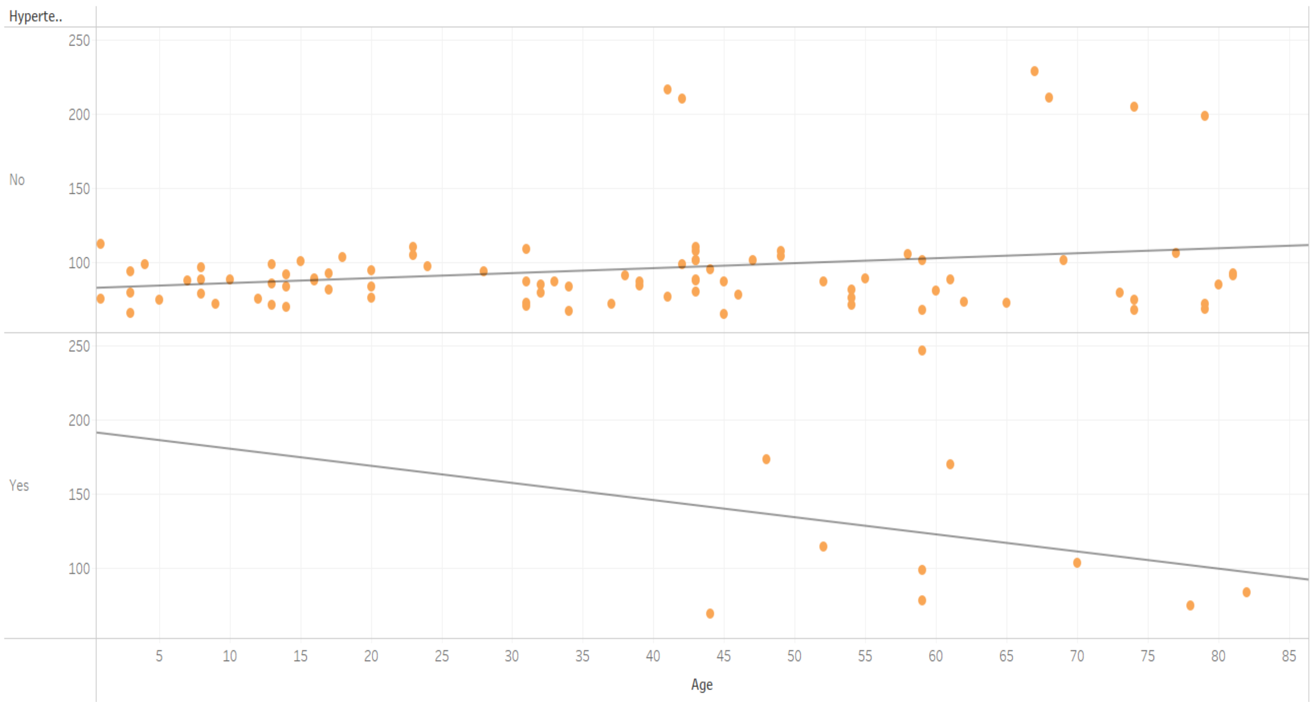


Fig 6.2.9 - Trendline of Age vs Hyper Tension

6.3 Cohort Analysis

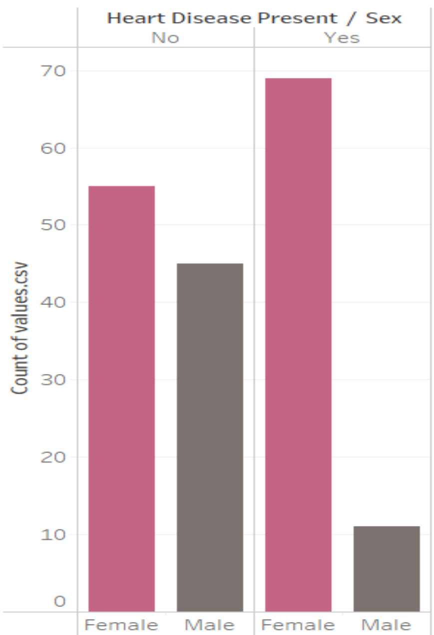


Fig 6.3.1 - Heart Disease present vs gender with patient name

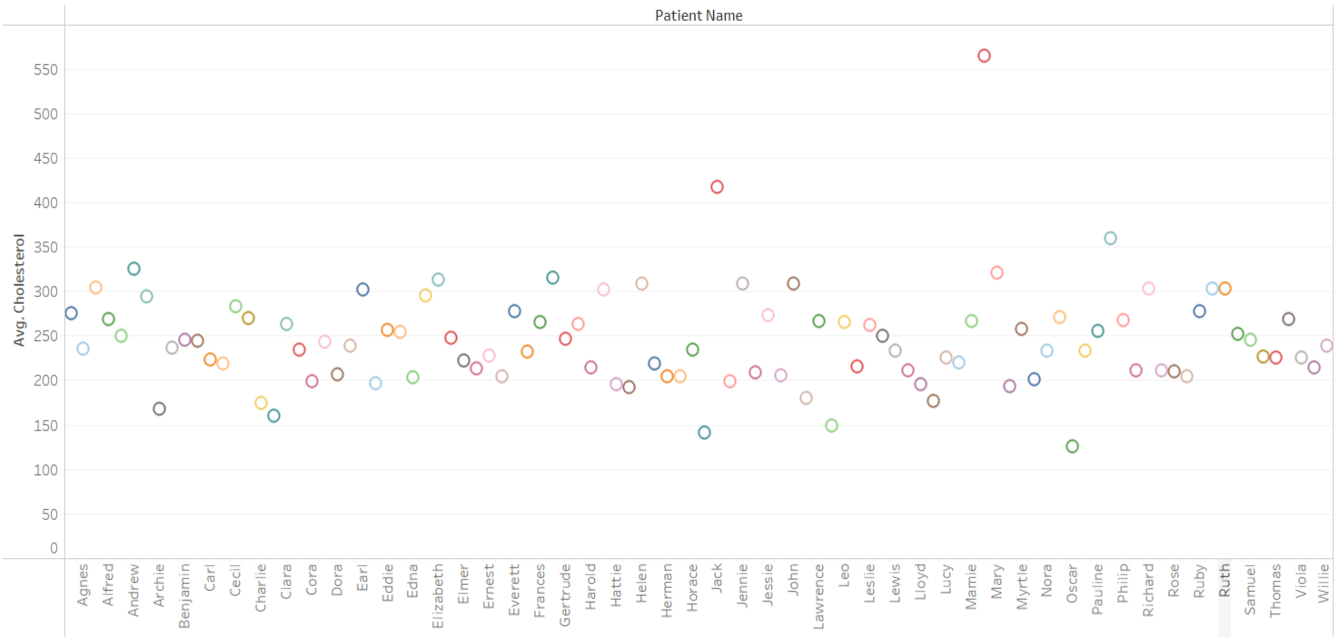


Fig 6.3.2 - Scatter plot of avg cholesterol vs patients

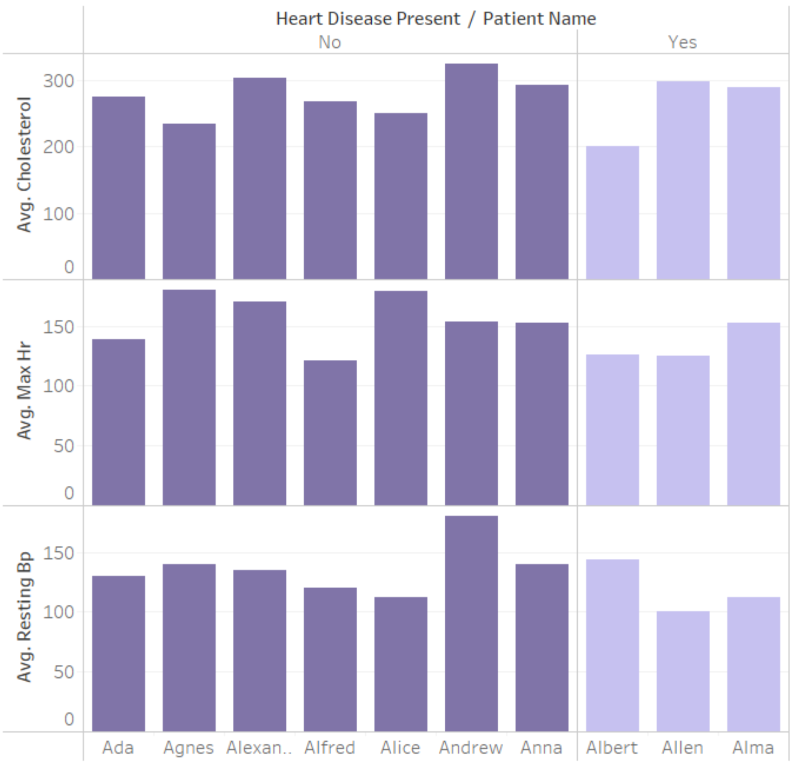


Fig 6.3.3 - Tree map of types of chest pain vs the presence of heart disease

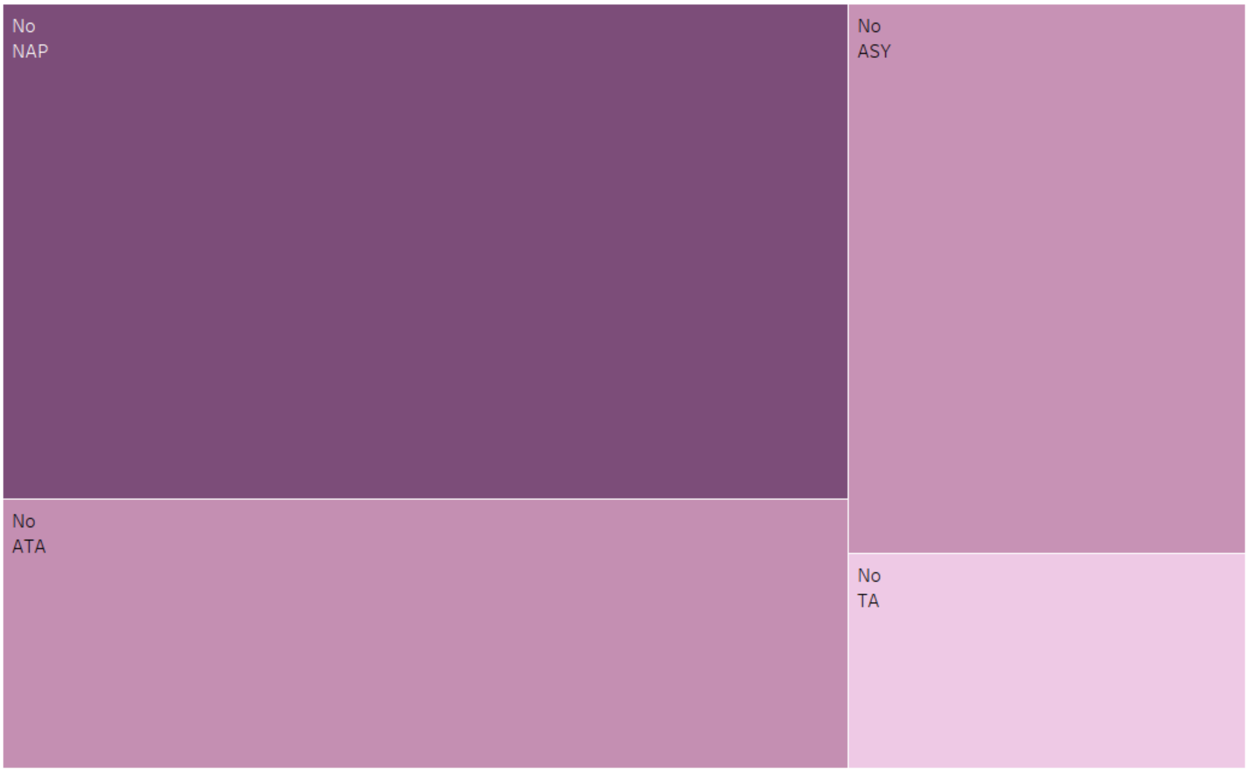


Fig 6.3.4

7. Machine Learning

7.1 Biological Factors

Reading the Dataset

```
data = pd.read_csv("heart.csv")
data.head()
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	No	Normal	172	N	0.0	Up	No
1	49	F	NAP	160	180	No	Normal	156	N	1.0	Flat	Yes
2	37	M	ATA	130	283	No	ST	98	N	0.0	Up	No
3	48	F	ASY	138	214	No	Normal	108	Y	1.5	Flat	Yes
4	54	M	NAP	150	195	No	Normal	122	N	0.0	Up	No

```
print(f"Shape of Dataframe is: {data.shape}")
```

```
Shape of Dataframe is: (303, 14)
```

Fig 7.1.1

```python
print('Datatype in Each Column')
pd.DataFrame(data.dtypes, columns=['Datatype']).rename_axis("Column Name")
```

Datatype in Each Column

Column Name	Datatype
age	int64
sex	int64
cp	int64
trestbps	int64
chol	int64
fbs	int64
restecg	int64
thalach	int64
exang	int64
oldpeak	float64
slope	int64
ca	int64
thal	int64
target	int64

Fig 7.1.2

```python
data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.31
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.61
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.00
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.00
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.00
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.00

Fig 7.1.3

Data Preprocessing

```
pd.DataFrame(data.isna().sum(), columns=["Null Values"]).rename_axis("Column Name")
```

Column Name	Null Values
age	0
sex	0
cp	0
trestbps	0
chol	0
fbs	0
restecg	0
thalach	0
exang	0
oldpeak	0
slope	0
ca	0
thal	0
target	0

Fig 7.1.4

Machine Learning

```
data.insert(0, 'id', range(1, 1 + len(data)))
data.head()
```

	id	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	1	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	2	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	3	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	4	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	5	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Splitting the data into train and test datasets

```
#Splitting the independent variables and target variable -stroke classification
X = data.drop(["id","target"], axis=1)
y = data["target"]
y= pd.DataFrame(y,columns=["target"])
```

Encoding categorical variables

```
def sexEncoder(df):
 labelEncoder = LabelEncoder()
 df["sex"] = labelEncoder.fit_transform(df["sex"])
 df.head()
#male-1
#female-0
```

```
sexEncoder(data)
```

Fig 7.1.5

Standardizing the data

```
numeric_cols = X.select_dtypes(["float64","int64"])
scaler = StandardScaler()
X[numeric_cols.columns] = scaler.fit_transform(X[numeric_cols.columns])
```

```
numeric_cols=X[numeric_cols.columns].round(2)
```

```
numeric_cols.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	0.95	0.68	1.97	0.76	-0.26	2.39	-1.01	0.02	-0.70	1.09	-2.27	-0.71	-2.15
1	-1.92	0.68	1.00	-0.09	0.07	-0.42	0.90	1.63	-0.70	2.12	-2.27	-0.71	-0.51
2	-1.47	-1.47	0.03	-0.09	-0.82	-0.42	-1.01	0.98	-0.70	0.31	0.98	-0.71	-0.51
3	0.18	0.68	0.03	-0.66	-0.20	-0.42	0.90	1.24	-0.70	-0.21	0.98	-0.71	-0.51
4	0.29	-1.47	-0.94	-0.66	2.08	-0.42	0.90	0.58	1.44	-0.38	0.98	-0.71	-0.51

```
categorical_vbles = X.select_dtypes("object")
X = pd.get_dummies(X, columns=categorical_vbles.columns)
```

```
 categorical_vbles=X.round(2)
```

```
categorical_vbles.shape
```

```
(303, 13)
```

```
data=pd.concat([categorical_vbles,y],axis=1)
data=data.dropna()
```

Fig 7.1.6

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
model_comparison = pd.DataFrame(columns=["Model","Accuracy Score"])
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
logit=LogisticRegression(solver = "liblinear",random_state=0)
logit.fit(X_train,y_train)
y_pred = logit.predict(X_test)
score = accuracy_score(y_pred, y_test)
print(f"Logistic Regression: {score}")
```

```
Logistic Regression: 0.8131868131868132
```

```
add_model={"Model": "LogisticRegression", "Accuracy Score": round(score,4)}
model_comparison = model_comparison.append(add_model, ignore_index=True)
```

K-nearest Neighbours

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn_model = knn.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
score_knn =accuracy_score(y_test, y_pred)
print(f"KNeighborsClassifier: {score_knn}")
```

```
KNeighborsClassifier: 0.8791208791208791
```

Fig 7.1.7

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(criterion='entropy', ccp_alpha=0.003)
dtc.fit(X_train,y_train)
y_Pred = dtc.predict(X_test)
score = accuracy_score(y_Pred, y_test)
print(f"DecisionTreeClassifier: {score}")
```

```
DecisionTreeClassifier: 0.7252747252747253
```

```
add_model={"Model": "DecisionTreeClassifier", "Accuracy Score": round(score,4)}
model_comparison = model_comparison.append(add_model, ignore_index=True)
```

Random Forest classifier

```
from sklearn.ensemble import RandomForestClassifier
randomforest = RandomForestClassifier(n_estimators=1000, random_state=30)
randomforest.fit(X_train, y_train)
y_pred = randomforest.predict(X_test)
score = accuracy_score(y_pred, y_test)
print(f"RandomForestClassifier: {score}")
```

```
RandomForestClassifier: 0.8131868131868132
```

```
add_model={"Model": "RandomForestClassifier", "Accuracy Score": round(score,4)}
model_comparison = model_comparison.append(add_model, ignore_index=True)
```

Fig 7.1.8

Models and Accuracy Scores

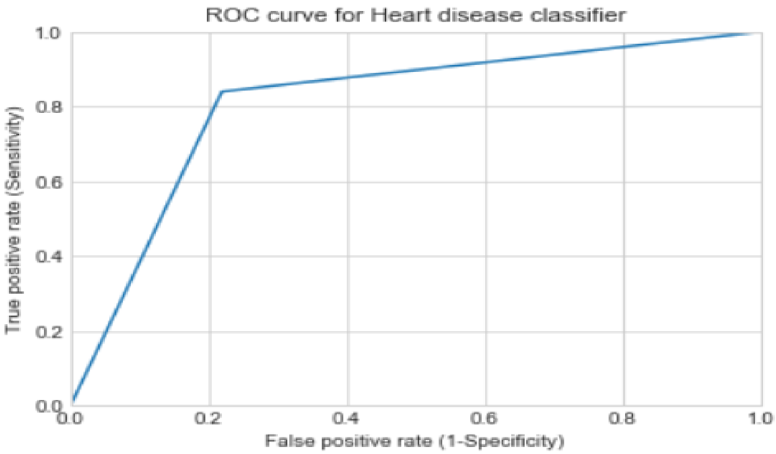
```
model_comparison
```

	Model	Accuracy Score
0	LogisticRegression	0.8132
1	KNeighborsClassifier	0.8132
2	DecisionTreeClassifier	0.7253
3	RandomForestClassifier	0.8132

Fig 7.1.9

ROC Curve

```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.plot(fpr,tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC curve for Heart disease classifier')
plt.xlabel('False positive rate (1-Specificity)')
plt.ylabel('True positive rate (Sensitivity)')
plt.grid(True)
```




```
import sklearn
sklearn.metrics.roc_auc_score(y_test,y_pred)
```

```
0.8102439024390243
```

Fig 7.1.10

7.2 Social Factors

Reading the Dataset

```
data = pd.read_csv("D:/Excel Sheets/Datastes/healthcare-dataset-stroke-data.csv")
data.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9No46	Male	67.0	No	Yes	Yes	Private	Urban	228.69	36.6	formerly smoked	Yes
1	51676	Female	61.0	No	No	Yes	Self-employed	Rural	202.21	NaN	never smoked	Yes
2	31112	Male	80.0	No	Yes	Yes	Private	Rural	105.92	32.5	never smoked	Yes
3	60182	Female	49.0	No	No	Yes	Private	Urban	171.23	34.4	smokes	Yes
4	1665	Female	79.0	Yes	No	Yes	Self-employed	Rural	174.12	24.0	never smoked	Yes

```
print(f"Shape of Dataframe is: {data.shape}")
```

```
Shape of Dataframe is: (5110, 12)
```

Fig 7.2.1

```python
print(f"Shape of Dataframe is: {data.shape}")
```

```
Shape of Dataframe is: (5110, 12)
```

```python
print('Datatype in Each Column')
pd.DataFrame(data.dtypes, columns=['Datatype']).rename_axis("Column Name")
```

```
Datatype in Each Column
```

Column Name	Datatype
id	object
gender	object
age	float64
hypertension	object
heart_disease	object
ever_married	object
work_type	object
Residence_type	object
avg_glucose_level	float64
bmi	float64
smoking_status	object
stroke	object

Fig 7.2.2

```python
data.describe()
```

	age	avg_glucose_level	bmi
count	5110.000000	5110.000000	4909.000000
mean	43.226614	106.147677	28.893237
std	22.612647	45.283560	7.854067
min	0.080000	55.120000	10.300000
25%	25.000000	77.245000	23.500000
50%	45.000000	91.885000	28.100000
75%	61.000000	114.090000	33.100000
max	82.000000	271.740000	97.600000

Fig 7.2.3

Data Preprocessing

```
pd.DataFrame(data.isna().sum(), columns=["Null Values"]).rename_axis("Column Name")
```

Column Name	Null Values
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

```
data['bmi'].fillna(data['bmi'].mean(), inplace=True)
```

```
other_index = data[data['gender'] =='Other'].index
data= data.drop(other_index)
```

Fig 7.2.4

```
data["smoking_status"].replace("Unknown", data["smoking_status"].mode().values[0], inplace=True)
```

```
pd.DataFrame(data.isna().sum(), columns=["Null Values"]).rename_axis("Column Name")
```

Column Name	Null Values
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

Fig 7.2.5

Machine Learning

Splitting the data into train and test datasets

```
#Splitting the independent variables and target variable -stroke classification
X = data.drop(["id","stroke"], axis=1)
y = data["stroke"]
y= pd.DataFrame(y,columns=["stroke"])
```

Encoding categorical variables

```
def genderEncoder(df):
 labelEncoder = LabelEncoder()
 df["gender"] = labelEncoder.fit_transform(df["gender"])
 df.head()
#male-1
#female-0
```

```
genderEncoder(data)
```

Fig 7.2.6

Standardizing the data

```
numeric_cols = X.select_dtypes(["float64","int64"])
scaler = StandardScaler()
X[numeric_cols.columns] = scaler.fit_transform(X[numeric_cols.columns])
```

```
numeric_cols=X[numeric_cols.columns].round(2)
```

```
numeric_cols.head()
```

	age	avg_glucose_level	bmi
0	1.05	2.71	1.00
1	0.79	2.12	-0.00
2	1.63	-0.00	0.47
3	0.26	1.44	0.72
4	1.58	1.50	-0.64

```
categorical_vbles = X.select_dtypes("object")
X = pd.get_dummies(X, columns=categorical_vbles.columns)
```

```
 categorical_vbles=X.round(2)
```

```
categorical_vbles.shape
```

```
(5109, 21)
```

Fig 7.2.7

```
data=pd.concat([categorical_vbles,y],axis=1)
data=data.dropna()
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
model_comparison = pd.DataFrame(columns=["Model","Accuracy Score"])
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
logit=LogisticRegression(solver = "liblinear",random_state=0)
logit.fit(X_train,y_train)
y_pred = logit.predict(X_test)
score = accuracy_score(y_pred, y_test)
print(f"Logistic Regression: {score}")
```

```
Logistic Regression: 0.9425962165688193
```

```
add_model={"Model": "LogisticRegression", "Accuracy Score": round(score,4)}
model_comparison = model_comparison.append(add_model, ignore_index=True)
```

K-nearest Neighbours

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn_model = knn.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
score_knn =accuracy_score(y_test, y_pred)
print(f"KNeighborsClassifier: {score_knn}")
```

```
KNeighborsClassifier: 0.9399869536855838
```

Fig 7.2.8

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(criterion='entropy', ccp_alpha=0.003)
dtc.fit(X_train,y_train)
y_Pred = dtc.predict(X_test)
score = accuracy_score(y_Pred, y_test)
print(f"DecisionTreeClassifier: {score}")
```

```
DecisionTreeClassifier: 0.9419439008480104
```

```
add_model={"Model": "DecisionTreeClassifier", "Accuracy Score": round(score,4)}
model_comparison = model_comparison.append(add_model, ignore_index=True)
```

Random Forest classifier

```
from sklearn.ensemble import RandomForestClassifier
randomforest = RandomForestClassifier(n_estimators=1000, random_state=30)
randomforest.fit(X_train, y_train)
y_pred = randomforest.predict(X_test)
score = accuracy_score(y_pred, y_test)
print(f"RandomForestClassifier: {score}")
```

```
RandomForestClassifier: 0.9419439008480104
```

```
add_model={"Model": "RandomForestClassifier", "Accuracy Score": round(score,4)}
model_comparison = model_comparison.append(add_model, ignore_index=True)
```

Fig 7.2.9

Models and Accuracy Scores

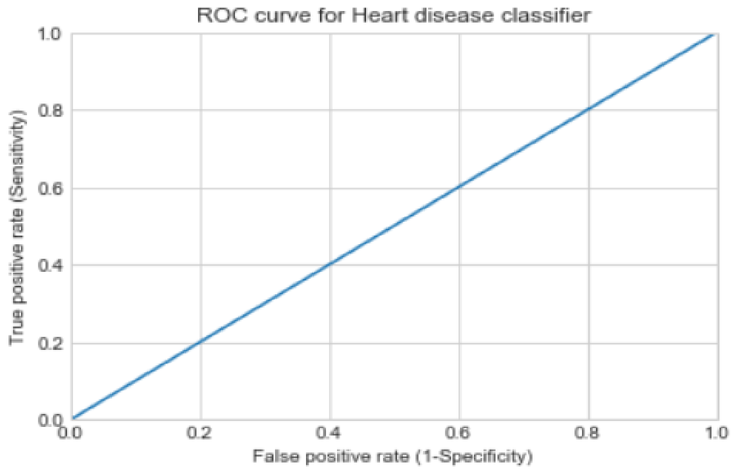
```
model_comparison
```

	Model	Accuracy Score
0	LogisticRegression	0.9426
1	KNeighborsClassifier	0.9426
2	DecisionTreeClassifier	0.9419
3	RandomForestClassifier	0.9419

Fig 7.2.10

ROC Curve

```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.plot(fpr,tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC curve for Heart disease classifier')
plt.xlabel('False positive rate (1-Specificity)')
plt.ylabel('True positive rate (Sensitivity)')
plt.grid(True)
```




```
import sklearn
sklearn.metrics.roc_auc_score(y_test,y_pred)
```

```
0.5
```

Fig 7.2.11

7.3 Cohort Analysis

Reading the Dataset

```
data = pd.read_csv("D:/Excel Sheets/Datastes/values.csv")
data.head()
```

	patient name	thalassemia	resting bp	chest_pain_type	fasting bs	cholesterol	depression	sex	age	max hr	exercise	heart_disease_present
0	John	normal	128	ATA	No	308	No	Female	45	170	No	No
1	William	normal	110	NAP	No	214	Yes	Male	54	158	No	No
2	James	normal	125	ASY	No	304	No	Female	77	162	Yes	Yes
3	George	reversible_defect	152	ASY	No	223	No	Female	40	181	No	Yes
4	Charles	reversible_defect	178	TA	No	270	Yes	Female	59	145	No	No

Fig 7.3.1

```
print('Datatype in Each Column')
pd.DataFrame(data.dtypes, columns=['Datatype']).rename_axis("Column Name")
```

```
Datatype in Each Column
```

Column Name	Datatype
patient name	object
thalassemia	object
resting bp	int64
chest_pain_type	object
fasting bs	object
cholesterol	int64
depression	object
sex	object
age	int64
max hr	int64
exercise	object
heart_disease_present	object

Fig 7.3.2

```
data.describe()
```

	resting bp	cholesterol	age	max hr
count	180.000000	180.000000	180.000000	180.000000
mean	131.311111	249.211111	54.811111	149.483333
std	17.010443	52.717969	9.334737	22.063513
min	94.000000	126.000000	29.000000	96.000000
25%	120.000000	213.750000	48.000000	132.000000
50%	130.000000	245.500000	55.000000	152.000000
75%	140.000000	281.250000	62.000000	166.250000
max	180.000000	564.000000	77.000000	202.000000

Fig 7.3.3

```
pd.DataFrame(data.isna().sum(), columns=["Null Values"]).rename_axis("Column Name")
```

	Null Values
Column Name	
patient name	0
thalassemia	0
resting bp	0
chest_pain_type	0
fasting bs	0
cholesterol	0
depression	0
sex	0
age	0
max hr	0
exercise	0
heart_disease_present	0

Fig 7.3.4

Machine Learning

Splitting the data into train and test datasets

```
data.insert(0, 'id', range(1, 1 + len(data)))
heart_disease_present = {'No': 0, 'Yes': 1}
data.heart_disease_present = [heart_disease_present[item] for item in data.heart_disease_present]
data.head()
```

	id	patient name	thalassemia	resting bp	chest_pain_type	fasting bs	cholesterol	depression	sex	age	max hr	exercise	heart_disease_present
0	1	John	normal	128	ATA	No	308	No	Female	45	170	No	0
1	2	William	normal	110	NAP	No	214	Yes	Male	54	158	No	0
2	3	James	normal	125	ASY	No	304	No	Female	77	162	Yes	1
3	4	George	reversible_defect	152	ASY	No	223	No	Female	40	181	No	1
4	5	Charles	reversible_defect	178	TA	No	270	Yes	Female	59	145	No	0

Splitting the data into train and test datasets

```
#Splitting the independent variables and target variable -stroke classification
X = data.drop(["id","heart_disease_present"], axis=1)
y = data["heart_disease_present"]
y= pd.DataFrame(y,columns=["heart_disease_present"])
```

Fig 7.3.5

Splitting the data into train and test datasets

```
#Splitting the independent variables and target variable -stroke classification
X = data.drop(["id","heart_disease_present"], axis=1)
y = data["heart_disease_present"]
y= pd.DataFrame(y,columns=["heart_disease_present"])
```

Encoding categorical variables

```
def sexEncoder(df):
 labelEncoder = LabelEncoder()
 df["sex"] = labelEncoder.fit_transform(df["sex"])
 df.head()
#male-1
#female-0
```

```
sexEncoder(data)
```

Standardizing the data

```
numeric_cols = X.select_dtypes(["float64","int64"])
scaler = StandardScaler()
X[numeric_cols.columns] = scaler.fit_transform(X[numeric_cols.columns])
```

```
numeric_cols=X[numeric_cols.columns].round(2)
```

```
numeric_cols.head()
```

Fig 7.3.6

	resting bp	cholesterol	age	max hr
0	-0.20	1.12	-1.05	0.93
1	-1.26	-0.67	-0.09	0.39
2	-0.37	1.04	2.38	0.57
3	1.22	-0.50	-1.59	1.43
4	2.75	0.40	0.45	-0.20

```
categorical_vbles = X.select_dtypes("object")
X = pd.get_dummies(X, columns=categorical_vbles.columns)
```

```
categorical_vbles=X.round(2)
```

```
categorical_vbles.shape
```

```
(180, 197)
```

```
data=pd.concat([categorical_vbles,y],axis=1)
data=data.dropna()
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
model_comparison = pd.DataFrame(columns=["Model","Accuracy Score"])
```

Fig 7.3.7

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
logit=LogisticRegression(solver = "liblinear",random_state=0)
logit.fit(X_train,y_train)
y_pred = logit.predict(X_test)
score = accuracy_score(y_pred, y_test)
print(f"Logistic Regression: {score}")
```

```
Logistic Regression: 0.8148148148148148
```

```
add_model={"Model": "LogisticRegression", "Accuracy Score": round(score,4)}
model_comparison = model_comparison.append(add_model, ignore_index=True)
```

K-nearest Neighbours

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn_model = knn.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
score_knn =accuracy_score(y_test, y_pred)
print(f"KNeighborsClassifier: {score_knn}")
```

```
KNeighborsClassifier: 0.7962962962962963
```

```
add_model={"Model": "KNeighborsClassifier", "Accuracy Score": round(score,4)}
model_comparison = model_comparison.append(add_model, ignore_index=True)
```

Fig 7.3.8

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(criterion='entropy', ccp_alpha=0.003)
dtc.fit(X_train,y_train)
y_Pred = dtc.predict(X_test)
score = accuracy_score(y_Pred, y_test)
print(f"DecisionTreeClassifier: {score}")
```

```
DecisionTreeClassifier: 0.6851851851851852
```

```
add_model={"Model": "DecisionTreeClassifier", "Accuracy Score": round(score,4)}
model_comparison = model_comparison.append(add_model, ignore_index=True)
```

Random Forest classifier

```
from sklearn.ensemble import RandomForestClassifier
randomforest = RandomForestClassifier(n_estimators=1000, random_state=30)
randomforest.fit(X_train, y_train)
y_pred = randomforest.predict(X_test)
score = accuracy_score(y_pred, y_test)
print(f"RandomForestClassifier: {score}")
```

```
RandomForestClassifier: 0.8333333333333334
```

```
add_model={"Model": "RandomForestClassifier", "Accuracy Score": round(score,4)}
model_comparison = model_comparison.append(add_model, ignore_index=True)
```

Fig 7.3.9

Models and Accuracy Scores

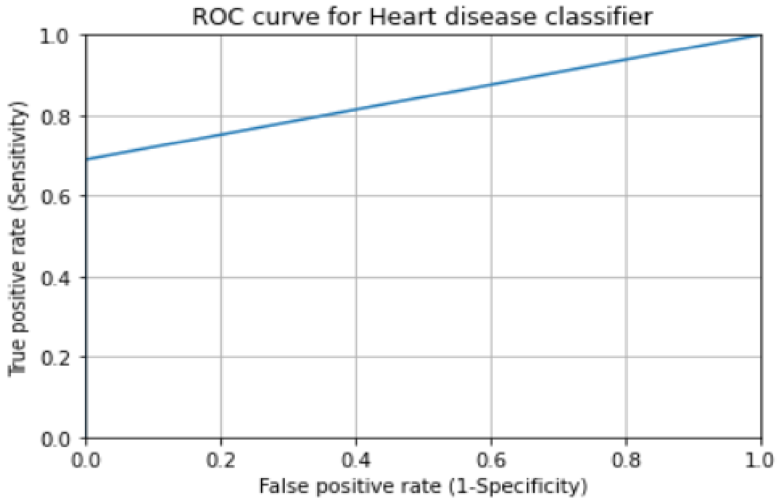
```
model_comparison
```

	Model	Accuracy Score
0	LogisticRegression	0.8148
1	KNeighborsClassifier	0.8148
2	DecisionTreeClassifier	0.6852
3	RandomForestClassifier	0.8333

Fig 7.3.10

ROC Curve

```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.plot(fpr,tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC curve for Heart disease classifier')
plt.xlabel('False positive rate (1-Specificity)')
plt.ylabel('True positive rate (Sensitivity)')
plt.grid(True)
```




```
import sklearn
sklearn.metrics.roc_auc_score(y_test,y_pred)
```

```
0.8448275862068966
```

Fig 7.3.11

8. Website

Fig 8.1 – Index page

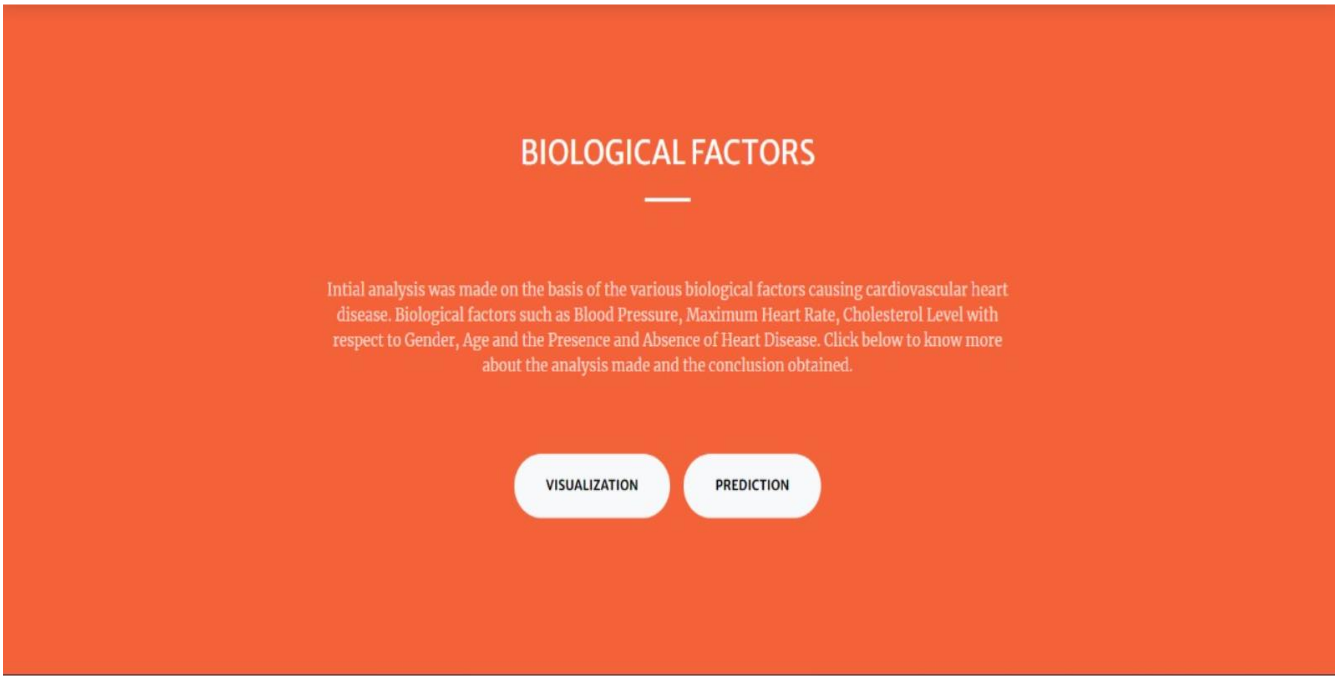


Fig 8.2 – Biological Factors

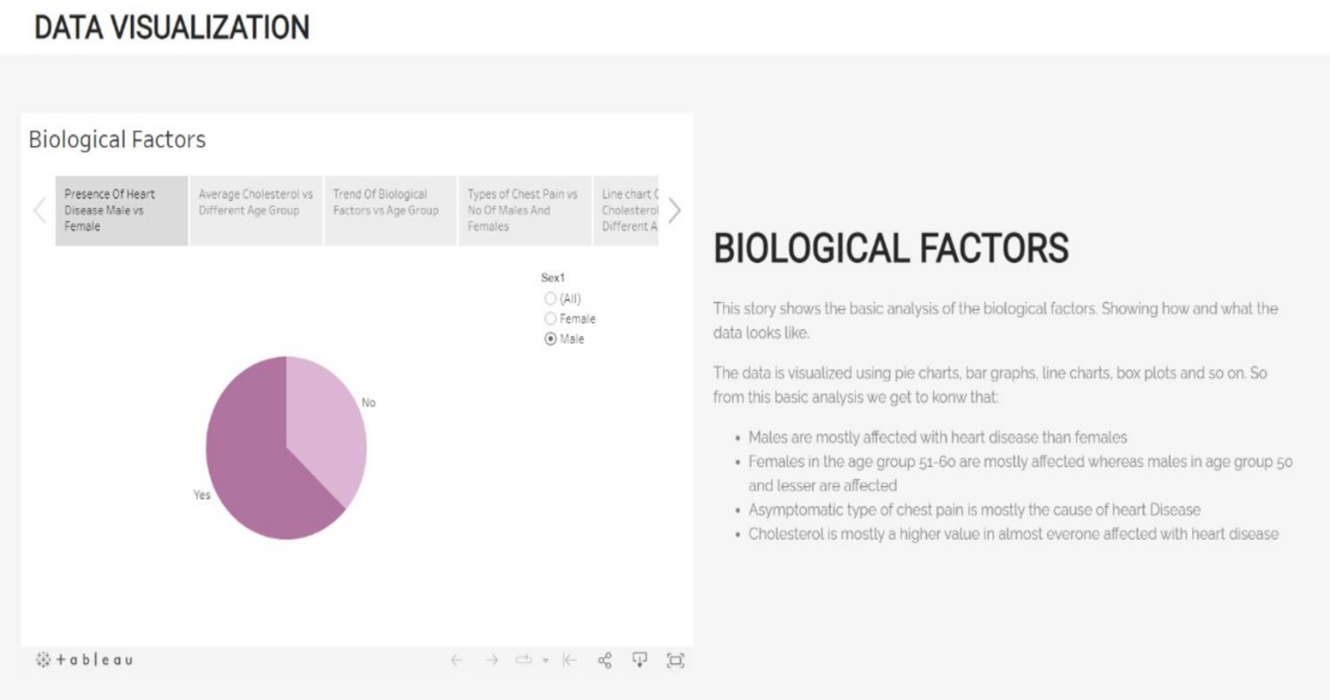


Fig 8.3 Data visualization

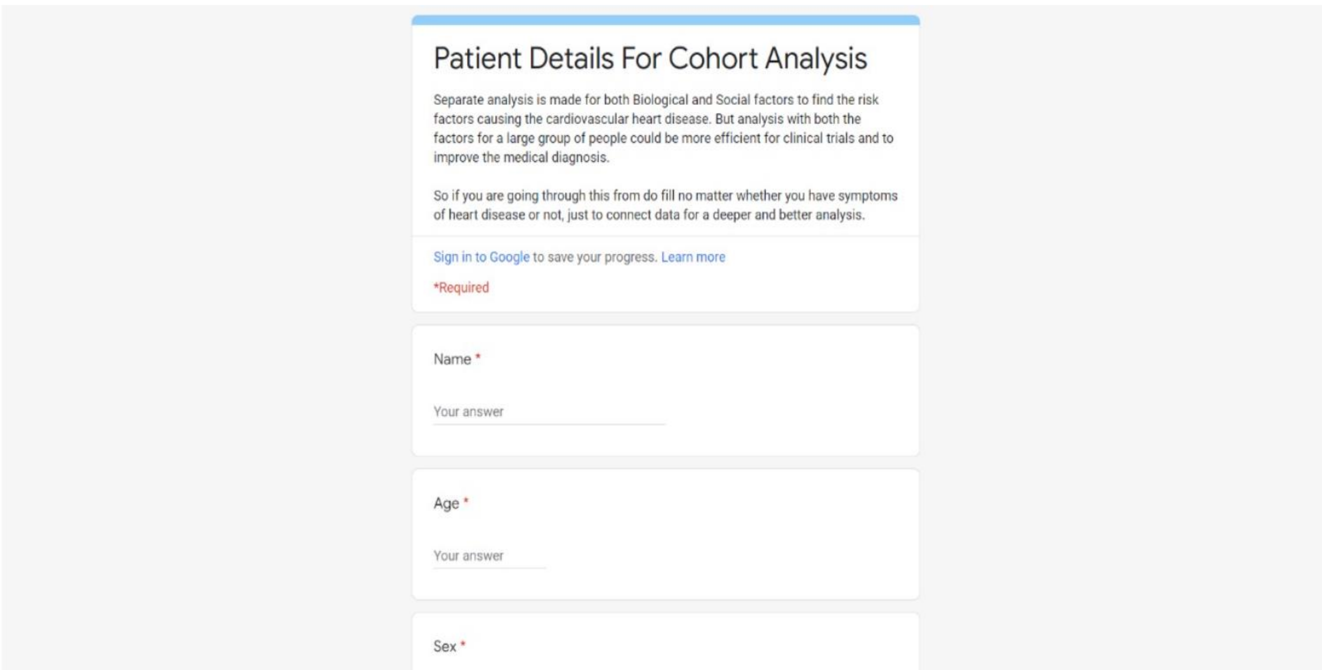


Fig 8.4 – Cohort form

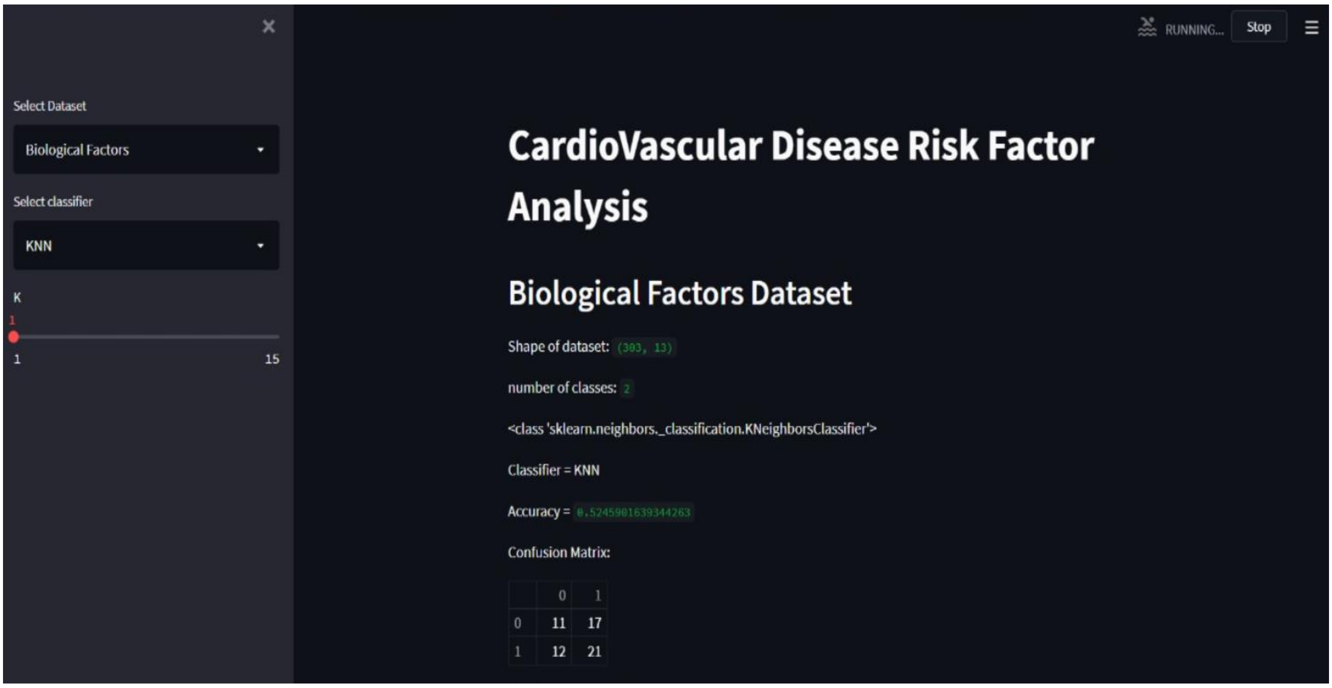


Fig 8.5 – Streamlit

Fig – 8.6 Overview

9. Project Work Repository

- GitHub Repository Link:

  https://github.com/2022-SRET-INT300/Cardiovascular-Disease

- Tableau Public Link:

  https://public.tableau.com/app/profile/joselyn.diana.cindrella#!/

10.Timeline

Day 11	11-Jan-2022	Hearts dataset collection
Day 12	12-Jan-2022	Healthcare and Patient wise dataset collection
Day 13	13-Jan-2022	Data cleaning
Day 14	14-Jan-2022	Exploratory hearts data analysis using R programming
Day 15	15-Jan-2022	Exploratory healthcare data analysis using R programming
Day 17	17-Jan-2022	Tried exploratory values data analysis using R programming
Day 18	18-Jan-2022	Tried exploratory values data analysis using R programming
Day 19	19-Jan-2022	Connecting R and Tableau
Day 20	20-Jan-2022	Connecting Python and Tableau
Day 21	21-Jan-2022	Learning how to do analysis using Tableau and R
Day 22	22-Jan-2022	Learning how to do analysis using Tableau and Python
Day 24	24-Jan-2022	Analysis - Most Likely to Have Heart Disease, Avg Cholesterol vs age grp, Major factors vs age group
Day 25	25-Jan-2022	Analysis - Relationship between gender and chest pain types, major factors age
Day 27	27-Jan-2022	Analysis - Social factors vs heart disease, social factors
Day 28	28-Jan-2022	Analysis - heart disease w.r.t gender & patient, Patient w.r.t Cholesterol
Day 29	29-Jan-2022	Analysis - Major factors vs random 10 ppl, Chest pain type
Day 31	31-Jan-2022	Dashboard creation
Day 32	1-Feb-2022	Story board creation and publishing it in tableau public
Day 33	2-Feb-2022	Project mentor meeting
Day 34	3-Feb-2022	Rectifying errors
Day 35	4-Feb-2022	PPT Preparation
Day 36	5-Feb-2022	Model 1st review
Day 38	7-Feb-2022	1st Review
Day 39	8-Feb-2022	1st Review
Day 40	9-Feb-2022	Analysis - Predictive modelling
Day 41	10-Feb-2022	Analysis - Trend line

Day 42	11-Feb-2022	Analysis - HR, BP, Cholesterol Vs Age
Day 43	12-Feb-2022	Analysis - Major factors using Box Plot with trend line
Day 45	14-Feb-2022	Project mentor meeting
Day 46	15-Feb-2022	Linear Regression - BP
Day 47	16-Feb-2022	Linear Regression - Glucose
Day 48	17-Feb-2022	Creating story and dashboard and publishing in tableau public
Day 49	18-Feb-2022	Social Factors trend lines
Day 50	19-Feb-2022	Social Factors - Data visualization
Day 52	21-Feb-2022	Social Factors - Density plots
Day 53	22-Feb-2022	Social Factors - Machine Learning models
Day 54	23-Feb-2022	Biological Factors - Data visualization
Day 55	24-Feb-2022	Biological Factors - Density plots
Day 56	25-Feb-2022	Biological Factors - Machine Learning models
Day 57	26-Feb-2022	PPT Preparation
Day 58	27-Feb-2022	2nd Review preparation
Day 59	28-Feb-2022	2nd Review
Day 61	2-Mar-2022	Website - Index page and nav bar creation
Day 62	3-Mar-2022	Website - Embedding Tableau page
Day 63	4-Mar-2022	Website - ML page
Day 64	5-Mar-2022	Working on website
Day 66	7-Mar-2022	Creating form for cohort analysis
Day 67	8-Mar-2022	Changes in visualization part
Day 68	9-Mar-2022	Learning stremlit
Day 69	10-Mar-2022	Learning stremlit
Day 70	11-Mar-2022	Loading dataset in stremlit
Day 71	12-Mar-2022	Trying to load stremlit app
Day 73	14-Mar-2022	Competing stremlit in web app
Day 74	15-Mar-2022	Competing stremlit in web app
Day 75	16-Mar-2022	Adding stremlit to website
Day 82	23-Mar-2022	Report Preparation
Day 83	24-Mar-2022	Report Preparation
Day 84	25-Mar-2022	Report Review
Day 85	26-Mar-2022	PPT Preparation
Day 87	28-Mar-2022	Model Presentation
Day 88	29-Mar-2022	Final Review Preparation
Day 89	30-Mar-2022	Final Review Preparation
Day 90	31-Mar-2022	Final Review

11.References

- https://www.kaggle.com/fedesoriano/heart-failure-prediction
- https://www.kaggle.com/praagnya/heart-disease-prediction
- https://visualbi.com/blogs/tableau/tableau-integration-r/
- https://sci-hub.hkvisa.net/10.1109/TENSYMP50017.2020.9230887
- https://www.analyticsvidhya.com/blog/2020/12/integrate-r-tableau-and-excel/