MA515:Foundations of Data Science
Project Report

Sanganwar Sourabh (2020CSB1121)
Akshat Toolaj Sinha (2020CSB1068)
Pratham Kundan (2020CSB1114)
Abhijith TR (2020CSB1062)

Final Year, B.Tech. Computer Science and Engineering,
IIT Ropar

Submitted to *Dr. Arun Kumar*, Department of Mathematics,
IIT Ropar

Car-Seats Analysis

1. Problem Statement:

To do exploratory data analysis on the CarSeats dataset. Use multiple linear regression to predict the Sales. Use lasso technique at different values of hyperparameter and check if some coefficients are zero.

2. Data Description:

- ➔ The Carseats dataset was allocated to me. The task was to predict Unit Sales at each location
- ➔ The dataset has a shape of (400,11).
- ➔ The shape indicates that we have 10 features, 1 target column and 400 data points.

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	9.50	138	73	11	276	120	Bad	42	17	True	True
1	11.22	111	48	16	260	83	Good	65	10	True	True
2	10.06	113	35	10	269	80	Medium	59	12	True	True
3	7.40	117	100	4	466	97	Medium	55	14	True	True
4	4.15	141	64	3	340	128	Bad	38	13	True	False
5	10.81	124	113	13	501	72	Bad	78	16	False	True
6	6.63	115	105	0	45	108	Medium	71	15	True	False
7	11.85	136	81	15	425	120	Good	67	10	True	True
8	6.54	132	110	0	108	124	Medium	76	10	False	False
9	4.69	132	113	0	131	124	Medium	76	17	False	True

Table 1: A sample of given dataset

3. Exploratory Data Analysis:

1. Dataset consists of following columns:

Sales: Unit sales (in thousands) at each location

CompPrice: Price charged by competitor at each location

Income: Community income level (in thousands of dollars)

Advertising: Local advertising budget for company at each location (in thousands of dollars)

Population: Population size in region (in thousands)

Price: Price company charges for car seats at each site

ShelveLoc: A factor with levels Bad, Good and Medium indicating the quality of the shelving location for the car seats at each site

Age: Average age of the local population

Education: Education level at each location

Urban: A factor with levels No and Yes to indicate whether the store is in an urban or rural location

US: A factor with levels No and Yes to indicate whether the store is in the US or not

All variables except ShelveLoc, Urban, US are numerical.

2. Description of features is obtained using df.describe().

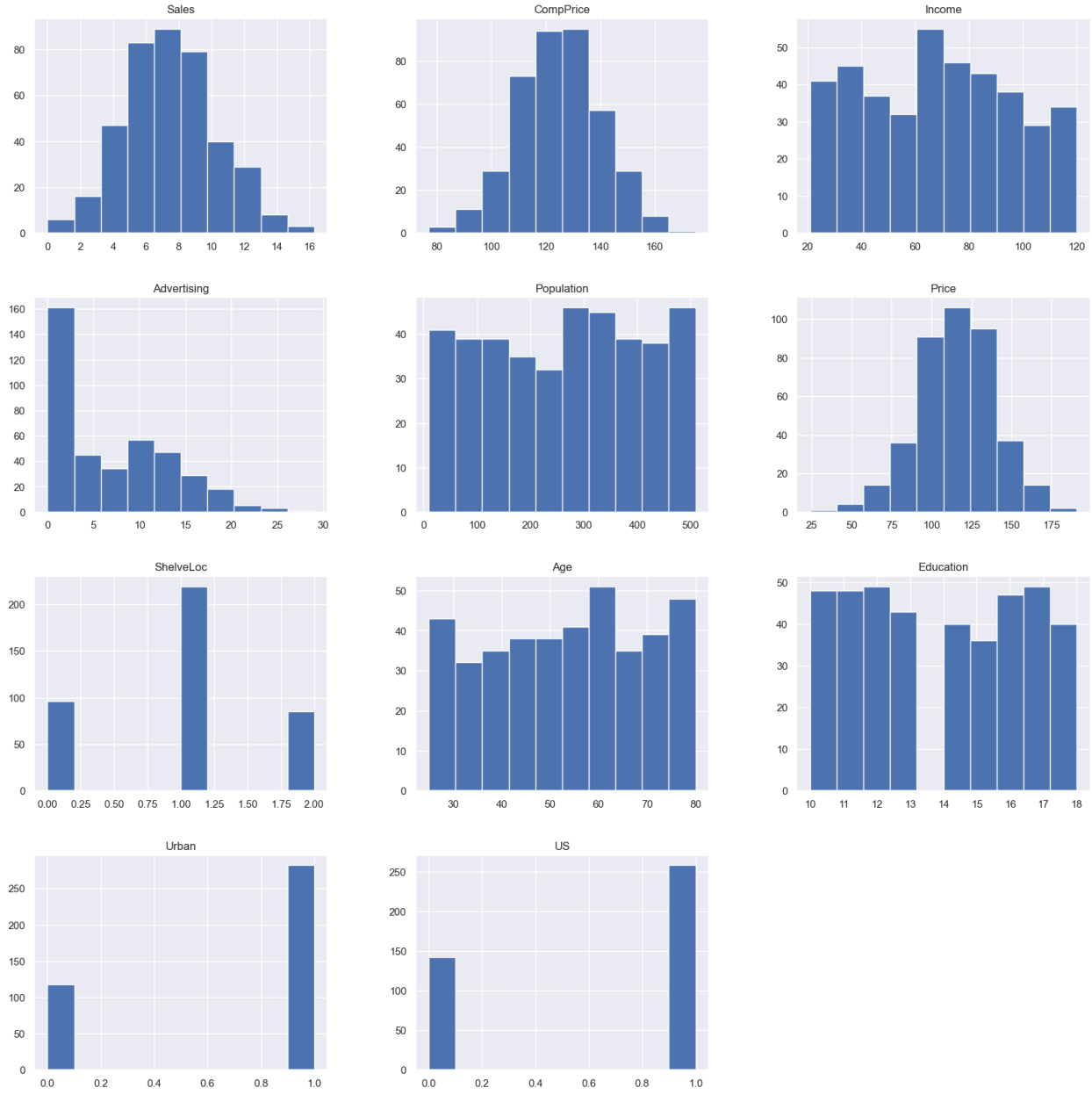
	Sales	CompPrice	Income	Advertising	Population	Price	Age	Education
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	7.496325	124.975000	68.657500	6.635000	264.840000	115.795000	53.322500	13.900000
std	2.824115	15.334512	27.986037	6.650364	147.376436	23.676664	16.200297	2.620528
min	0.000000	77.000000	21.000000	0.000000	10.000000	24.000000	25.000000	10.000000
25%	5.390000	115.000000	42.750000	0.000000	139.000000	100.000000	39.750000	12.000000
50%	7.490000	125.000000	69.000000	5.000000	272.000000	117.000000	54.500000	14.000000
75%	9.320000	135.000000	91.000000	12.000000	398.500000	131.000000	66.000000	16.000000
max	16.270000	175.000000	120.000000	29.000000	509.000000	191.000000	80.000000	18.000000

Table 2: Data description for numerical features

	ShelveLoc	Urban	US
count	400	400	400
unique	3	2	2
top	Medium	True	True
freq	219	282	258

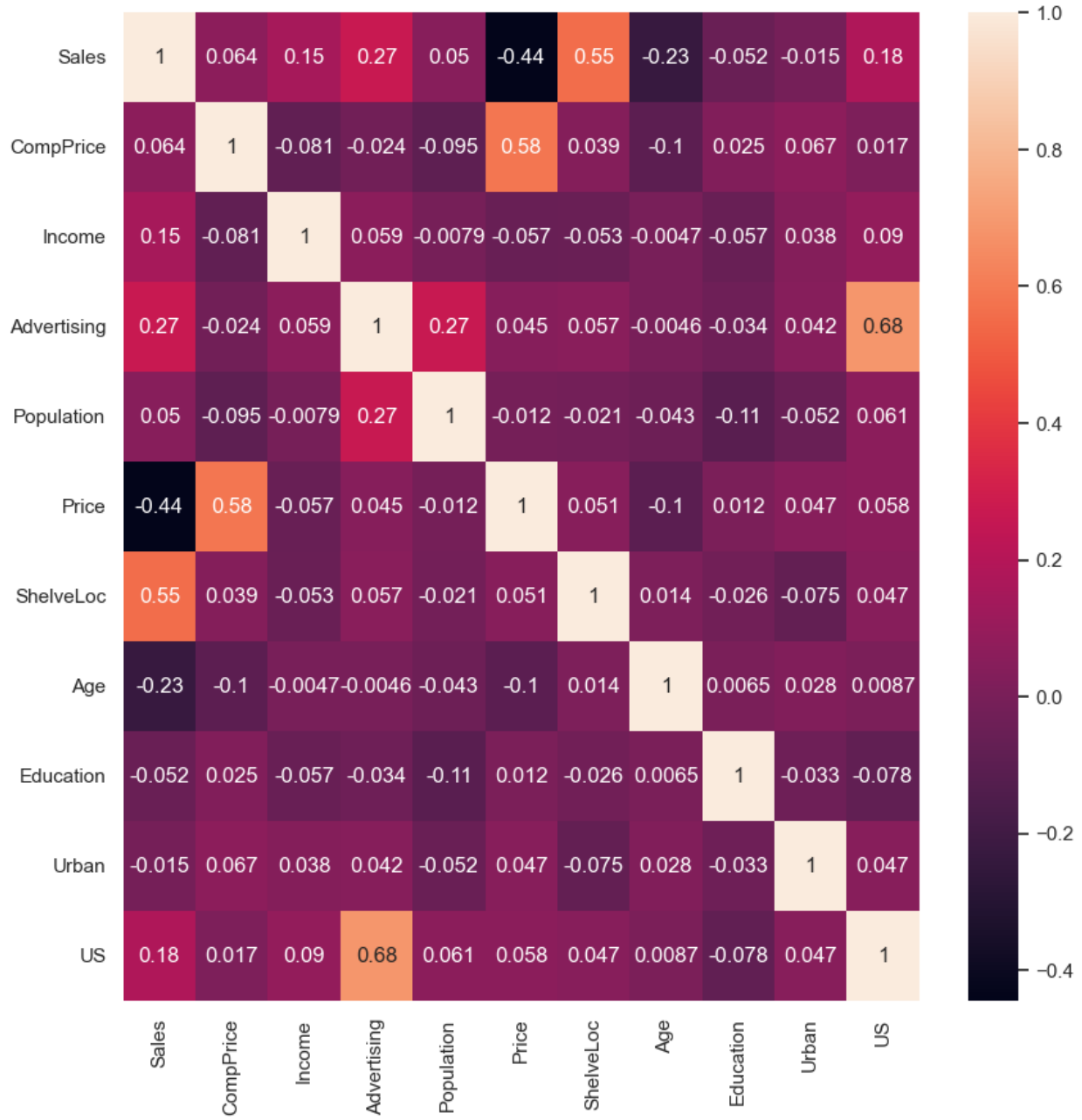
Table 3: Data description for Boolean/ Categorical Features

3. Converted boolean and categorical values to numerical. True as 1 and False as 0. And in ShelveLoc, Low as 0, medium as 1, high as 2.
4. Also we have plotted a histogram for each feature to understand the distribution of each feature.



Plot 1: Histograms of features distribution

5. We have plotted the values of correlation between features using df.corr()



Plot 2: Correlation plot

6. Then I have extracted X and y from the dataframe. Y is the sales column while X consists of remaining columns.

4. Split:

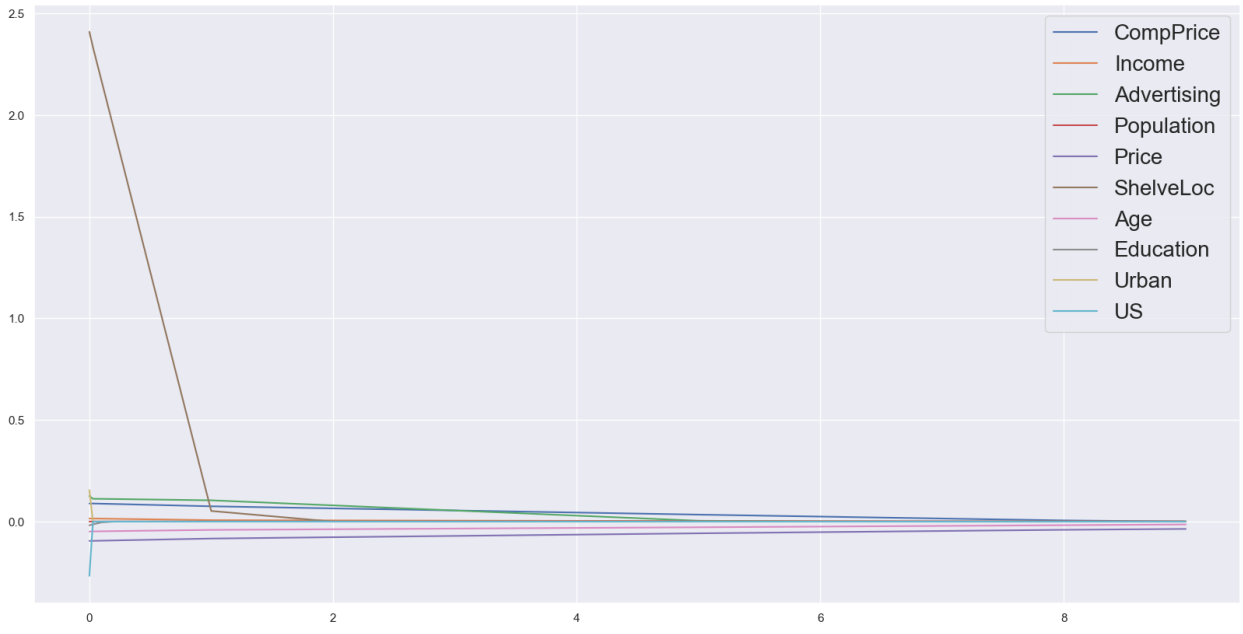
1. Here I have splitted the dataset into a training set and test set.
2. 80% of the data points are in the training set and remaining 20% in the test.

5. Multiple Linear Regression

1. Now, I have trained our model using LinearRegression() from sklearn on training data split, and tested using the test split. I got a Means Squared Error of `1.2140407850121935` and an R^2 Score of `0.8773225601508227`.
2. I have also tried training model on normalized features and obtained Means Squared Error and R^2 Score as `1.214040785012195` and `0.8773225601508226`
3. We can observe that there is no significant change between the both models. So feature normalization is not required.

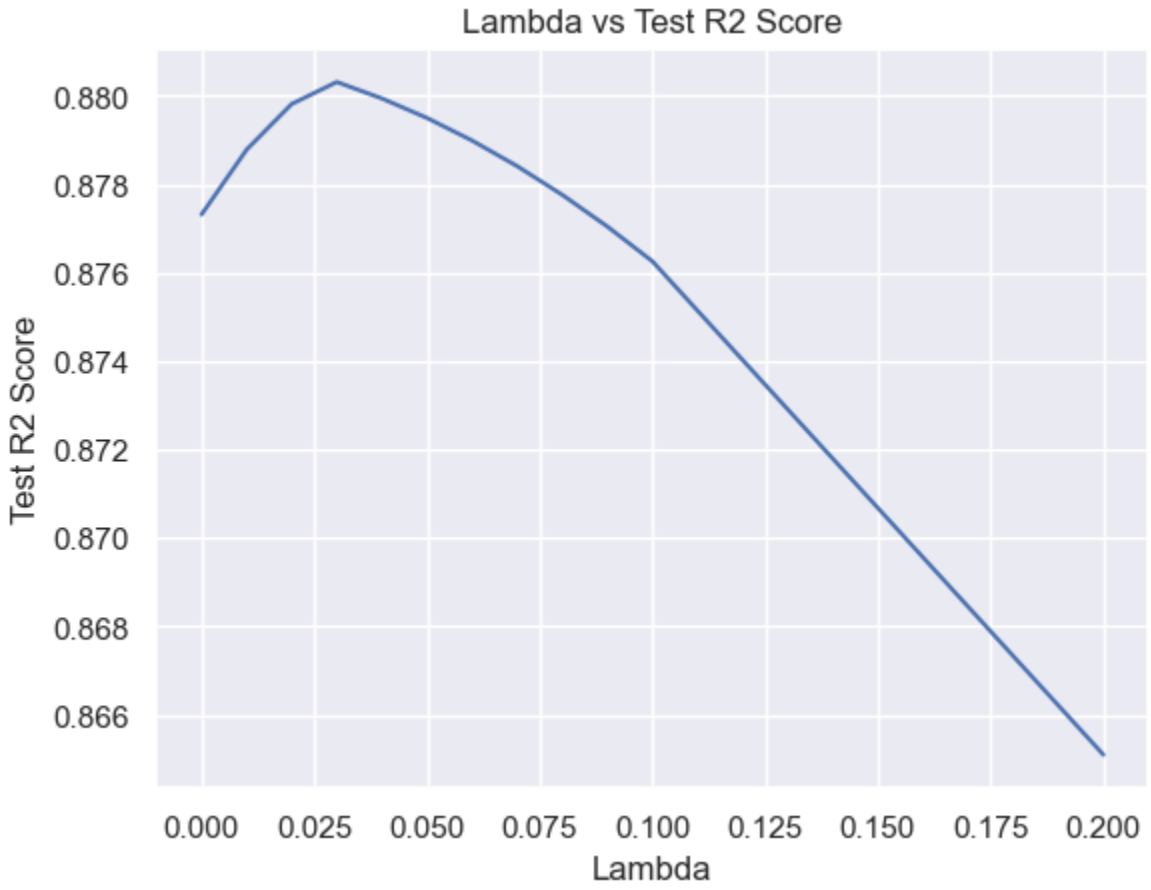
6. Lasso Multiple Linear Regression

1. I have trained LinearRegression models with Lasso on multiple lambda values.
2. US and Urban variable coefficients became 0 at lambda = 0.03, Education coefficient became 0 at lambda = 0.2. Remaining variables are also decreasing with increase in lambda.



Plot 3: Lamda vs coefficients of X.

3. I have observed that R^2 score was decreasing for train split while Increased for test split till lambda = 0.03 then decreases.



Plot 4: Lambda vs R^2 score

4. Best model was obtained at lambda = 0.03 with highest R^ squared of `0.8803189161152304`.

7. Comparison of Linear Regression vs Regression with Lasso:

1. Regular linear regression model is giving the highest training accuracy. In Lasso regression training accuracy was decreasing with increase in Lambda but testing accuracy increases to an extent then decreases. This is because in regression without lasso, we can observe that the weights value is large and the model is overfitting on training data split. Hence testing data is giving less R^2 score. Lasso limits the coefficient value by penalizing. So overfitting is reduced. We can observe the same. Regular regression gives 0.877 R^2 score while Lasso with 0.03 gives 0.88 R^2 score.

2. Hence Regression with lasso performs better than regular regression. Also we can see that US and Urban variable coefficients are 0 for lasso. This implies Lasso can predict better with fewer variables.

Cancer Analysis

1. Problem Statement:

Use a decision tree & random forest to classify whether the cancer is benign or malignant. Give a detailed comparative analysis

2. Data Description:

- The dataset has 30 Input features and 569 samples of patients
- The target variable is Diagnosis which is binary (‘M’ or ‘B’)
- Features include radius, smoothness, symmetry, concavity etc.

	Diagnosis	Radius (mean)	Texture (mean)	Perimeter (mean)	Area (mean)	Smoothness (mean)	Compactness (mean)	Concavity (mean)	Concave points (mean)	Symmetry (mean)	Fractal dimension (mean)	Radius (se)	Texture (se)	Perimeter (se)	Area (se)	Smoothness (se)	Compactness (se)
0	B	13.540	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.047810	0.1885	0.05766	0.2699	0.7886	2.058	23.560	0.008462	0.014600
1	B	13.080	15.71	85.63	520.0	0.10750	0.12700	0.04568	0.031100	0.1967	0.06811	0.1852	0.7477	1.383	14.670	0.004097	0.018980
2	B	9.504	12.44	60.34	273.9	0.10240	0.06492	0.02956	0.020760	0.1815	0.06905	0.2773	0.9768	1.909	15.700	0.009606	0.014320
3	B	13.030	18.42	82.61	523.8	0.08983	0.03766	0.02562	0.029230	0.1467	0.05863	0.1839	2.3420	1.170	14.160	0.004352	0.004899
4	B	8.196	16.84	51.71	201.9	0.08600	0.05943	0.01588	0.005917	0.1769	0.06503	0.1563	0.9567	1.094	8.205	0.008968	0.016460
5	B	12.050	14.63	78.04	449.3	0.10310	0.09092	0.06592	0.027490	0.1675	0.06043	0.2636	0.7294	1.848	19.870	0.005488	0.014270
6	B	13.490	22.30	86.91	561.0	0.08752	0.07698	0.04751	0.033840	0.1809	0.05718	0.2338	1.3530	1.735	20.200	0.004455	0.013820
7	B	11.760	21.60	74.72	427.9	0.08637	0.04966	0.01657	0.011150	0.1495	0.05888	0.4062	1.2100	2.635	28.470	0.005857	0.009758
8	B	13.640	16.34	87.21	571.8	0.07685	0.06059	0.01857	0.017230	0.1353	0.05953	0.1872	0.9234	1.449	14.550	0.004477	0.011770
9	B	11.940	18.24	75.71	437.6	0.08261	0.04751	0.01972	0.013490	0.1868	0.06110	0.2273	0.6329	1.520	17.470	0.007210	0.008380

Concavity (se)	Concave points (se)	Symmetry (se)	Fractal dimension (se)	Radius (worst)	Texture (worst)	Perimeter (worst)	Area (worst)	Smoothness (worst)	Compactness (worst)	Concavity (worst)	Concave points (worst)	Symmetry (worst)	Fractal dimension (worst)
0.02387	0.013150	0.01980	0.002300	15.110	19.26	99.70	711.2	0.14400	0.17730	0.23900	0.12880	0.2977	0.07259
0.01698	0.006490	0.01678	0.002425	14.500	20.49	96.09	630.5	0.13120	0.27760	0.18900	0.07283	0.3184	0.08183
0.01985	0.014210	0.02027	0.002968	10.230	15.66	65.13	314.9	0.13240	0.11480	0.08867	0.06227	0.2450	0.07773
0.01343	0.011640	0.02671	0.001777	13.300	22.81	84.46	545.9	0.09701	0.04619	0.04833	0.05013	0.1987	0.06169
0.01588	0.005917	0.02574	0.002582	8.964	21.96	57.26	242.2	0.12970	0.13570	0.06880	0.02564	0.3105	0.07409
0.02322	0.005660	0.01428	0.002422	13.760	20.70	89.88	582.6	0.14940	0.21560	0.30500	0.06548	0.2747	0.08301
0.02095	0.011840	0.01641	0.001956	15.150	31.82	99.00	698.8	0.11620	0.17110	0.22820	0.12820	0.2871	0.06917
0.01168	0.007445	0.02406	0.001769	12.980	25.72	82.98	516.5	0.10850	0.08615	0.05523	0.03715	0.2433	0.06563
0.01079	0.007956	0.01325	0.002551	14.670	23.19	96.08	656.7	0.10890	0.15820	0.10500	0.08586	0.2346	0.08025
0.01311	0.008000	0.01996	0.002635	13.100	21.33	83.67	527.2	0.11440	0.08906	0.09203	0.06296	0.2785	0.07408

Table-1: Sample of Dataset

3. Exploratory Data Analysis:

- No Null values are present in the dataset
- All columns are float64 except the target variable
- Mapped ‘M’ to 1 and ‘B’ to 0 for convenience.

```
Diagnosis                 object
Radius (mean)            float64
Texture (mean)           float64
Perimeter (mean)         float64
Area (mean)              float64
Smoothness (mean)        float64
Compactness (mean)       float64
Concavity (mean)         float64
Concave points (mean)    float64
Symmetry (mean)          float64
Fractal dimension (mean) float64
Radius (se)              float64
Texture (se)             float64
Perimeter (se)           float64
Area (se)                float64
Smoothness (se)          float64
Compactness (se)         float64
Concavity (se)           float64
Concave points (se)      float64
Symmetry (se)            float64
Fractal dimension (se)   float64
Radius (worst)           float64
Texture (worst)          float64
Perimeter (worst)        float64
Area (worst)             float64
Smoothness (worst)       float64
Compactness (worst)      float64
Concavity (worst)        float64
Concave points (worst)   float64
Symmetry (worst)         float64
Fractal dimension (worst) float64
dtype: object
```

Table-2: Dataset dtypes

```
Diagnosis                 0
Radius (mean)             0
Texture (mean)            0
Perimeter (mean)          0
Area (mean)               0
Smoothness (mean)         0
Compactness (mean)        0
Concavity (mean)          0
Concave points (mean)     0
Symmetry (mean)           0
Fractal dimension (mean)  0
Radius (se)               0
Texture (se)              0
Perimeter (se)            0
Area (se)                 0
Smoothness (se)           0
Compactness (se)          0
Concavity (se)            0
Concave points (se)       0
Symmetry (se)             0
Fractal dimension (se)    0
Radius (worst)            0
Texture (worst)           0
Perimeter (worst)         0
Area (worst)              0
Smoothness (worst)        0
Compactness (worst)       0
Concavity (worst)         0
Concave points (worst)    0
Symmetry (worst)          0
Fractal dimension (worst) 0
dtype: int64
```

Table-3: Dataset null values

- The dataset is described below

	Diagnosis	Radius (mean)	Texture (mean)	Perimeter (mean)	Area (mean)	Smoothness (mean)	Compactness (mean)	Concavity (mean)	Concave points (mean)	Symmetry (mean)	Fractal dimension (mean)	Radius (se)	Texture (se)	Perimeter (se)	Area (se)
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	0.372583	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	0.405172	1.216853	2.866059	40.337079
std	0.483918	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	0.277313	0.551648	2.021855	45.491006
min	0.000000	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	0.111500	0.360200	0.757000	6.802000
25%	0.000000	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	0.232400	0.833900	1.606000	17.850000
50%	0.000000	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	0.324200	1.108000	2.287000	24.530000
75%	1.000000	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	0.478900	1.474000	3.357000	45.190000
max	1.000000	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	2.873000	4.885000	21.980000	542.200000

Smoothness (se)	Compactness (se)	Concavity (se)	Concave points (se)	Symmetry (se)	Fractal dimension (se)	Radius (worst)	Texture (worst)	Perimeter (worst)	Area (worst)	Smoothness (worst)	Compactness (worst)	Concavity (worst)	Concave points (worst)	Symmetry (worst)	Fractal dimension (worst)
569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
0.007041	0.025478	0.031894	0.011796	0.020542	0.003795	16.269190	25.677223	107.261213	880.583128	0.132369	0.254265	0.272188	0.114606	0.290076	0.083946
0.003003	0.017908	0.030186	0.006170	0.008266	0.002646	4.833242	6.146258	33.602542	569.356993	0.022832	0.157336	0.208624	0.065732	0.061867	0.018061
0.001713	0.002252	0.000000	0.000000	0.007882	0.000895	7.930000	12.020000	50.410000	185.200000	0.071170	0.027290	0.000000	0.000000	0.156500	0.055040
0.005169	0.013080	0.015090	0.007638	0.015160	0.002248	13.010000	21.080000	84.110000	515.300000	0.116600	0.147200	0.114500	0.064930	0.250400	0.071460
0.006380	0.020450	0.025890	0.010930	0.018730	0.003187	14.970000	25.410000	97.660000	686.500000	0.131300	0.211900	0.226700	0.099930	0.282200	0.080040
0.008146	0.032450	0.042050	0.014710	0.023480	0.004558	18.790000	29.720000	125.400000	1084.000000	0.146000	0.339100	0.382900	0.161400	0.317900	0.092080
0.031130	0.135400	0.396000	0.052790	0.078950	0.029840	36.040000	49.540000	251.200000	4254.000000	0.222600	1.058000	1.252000	0.291000	0.663800	0.207500

Table-4: Column metrics

- Distribution of target variable is given below

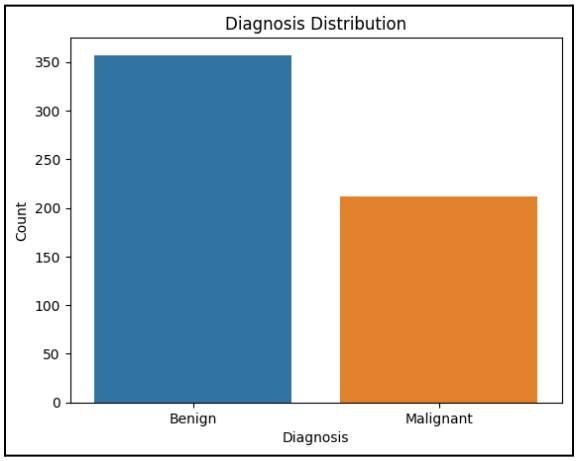


Fig-1: Target variable distribution

- Correlation matrix is given below, There are many highly corrrelated variables in the dataset which will area, perimeter, radius. This is obvious since perimeters and radius are determinitic functions of radius. Removing them make complexity of data lower.
- Data distribution of most columns looks like normal or log-normal distribution.
- The above steps are performed for visualization purpose.
- Data augmentations step is not required as decision tree and random forest does not require data to be normally distributed.
- 10 principal components are able to explain about 95% of variance in data
- Feature reduction is also not applied as number of columns is only 30, which is managable considering rows are also ~500.

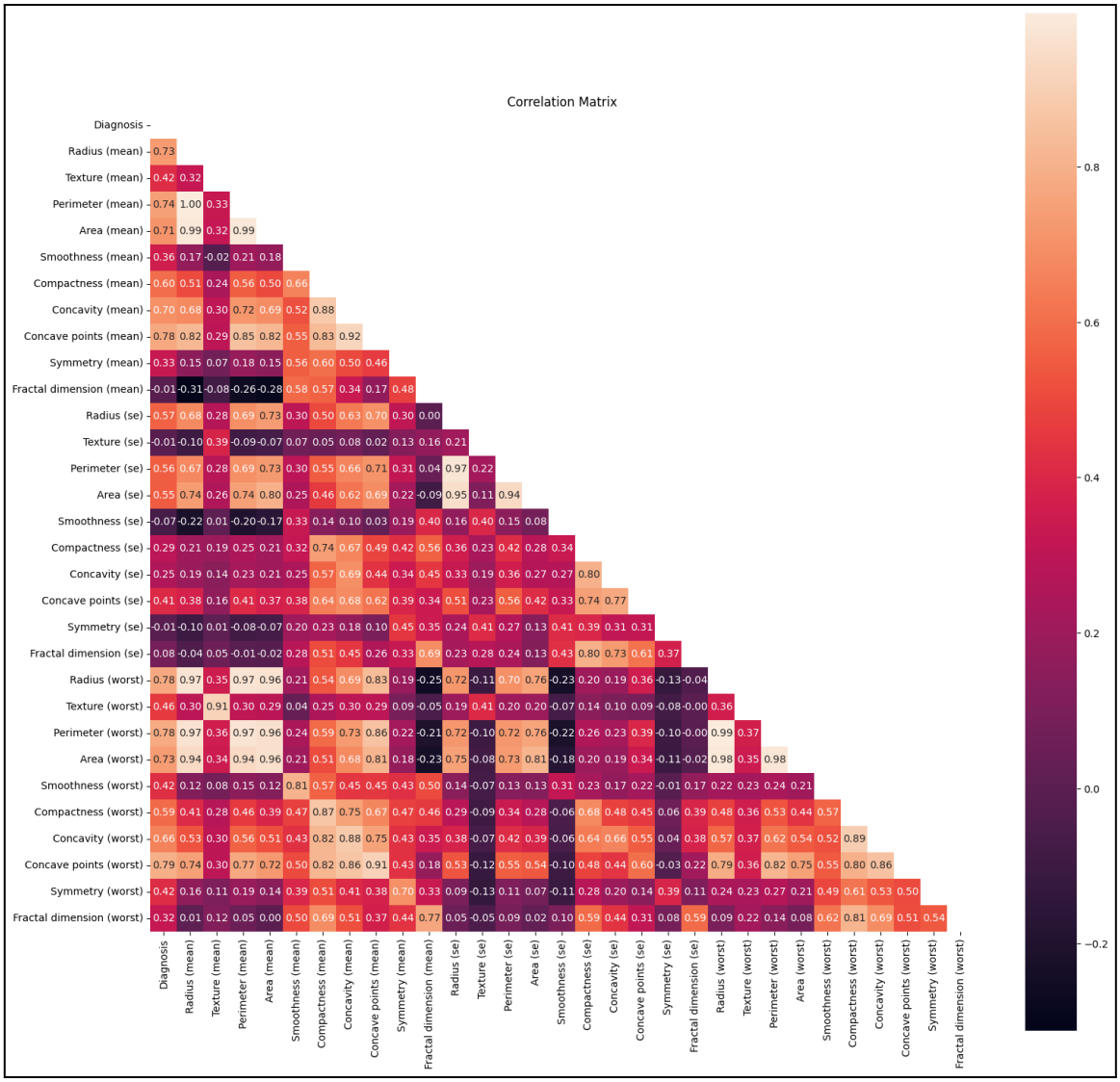


Fig-2: Correlation Matrix

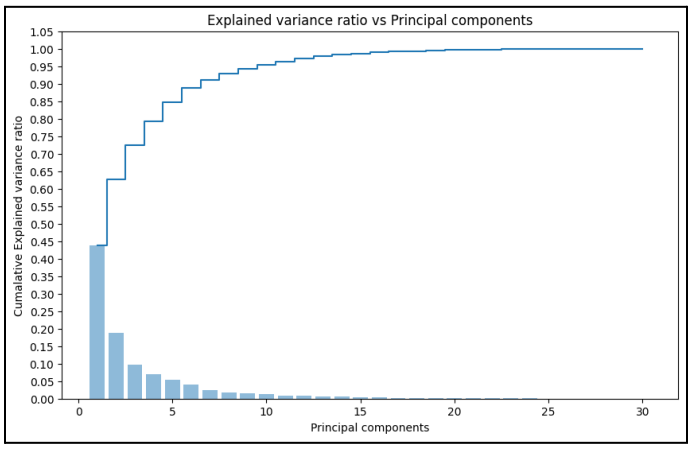


Fig-3: Cumulative Explained Variance

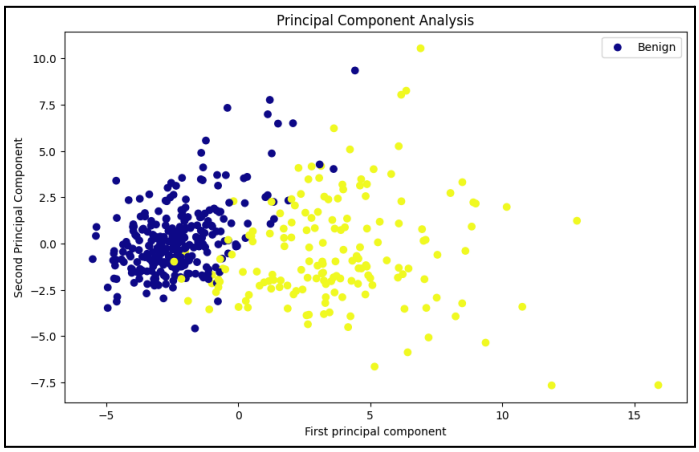


Fig-4 : PC1 vs PC2

4. Split:

We have splited the dataset into 75% training and 25% testing. This is an appropriate Splitting since we have very less number of data points.

5. Models:

Decision Tree and Random forest contains a lot of hyperparameters, so to tune hyperparameters, we have used GridSearchCV to determine optimal parameters from a set of values. GridSearchCV also contains K-fold cross validation internally and we have K=5

Below are specified ranges to GridSearchCV

1. max_depth : [4,6]
2. min_samples_leaf: [4,6]
3. min_samples_split: [4,6]
4. Criterion: [gini, entropy]
5. n_estimators: {50,100} (random forest only)

A brief explanation of above parameters are given below

1. max_depth: Maximum depth allowed in an individual tree
2. min_samples_leaf: Minimum samples required to be at a leaf node
3. min_samples_split: Minimum samples required to split a node
4. Criterion: Function to measure quality of split
5. n_estimations: Number of trees in whole forest.

We have calculated two metrics which are accuracy and F1 score.

F1 Score = 2*Precision*Recall / (Precision + Recall)

6. Decision Tree:

a. Used inbuilt DecisionTreeClassifier base model from sklearn.tree
b. Optimal parameters are max_depth=4, min_samples_leaf=6, min_samples_split=4, criterion=entropy
c. Accuracy for Predicting Yes: 48/50 = 96%
d. Accuracy for Predicting No: 91/93 = 97.84%

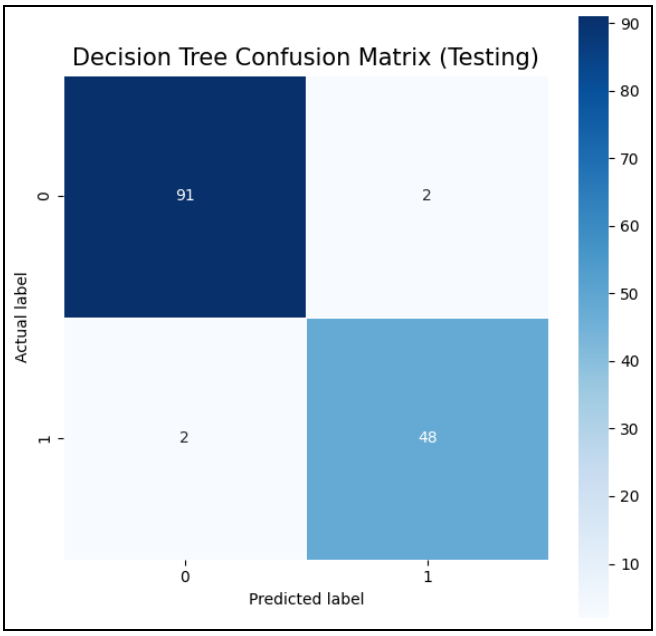


Fig-5 : Decision Tree Confusion matrix

7. Random Forest:

a. Used inbuilt RandomForestClassifier base model from sklearn.tree
b. Optimal parameters are max_depth=6, min_samples_leaf=4, min_samples_split=4, criterion=entropy
c. Accuracy for Predicting Yes: 50/50 = 100%
d. Accuracy for Predicting No: 92/93 = 98.92%

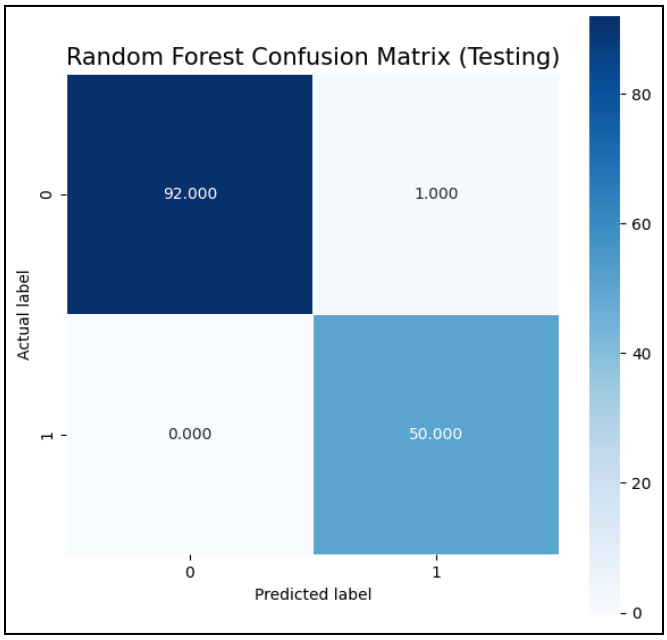


Fig-5 : Random Forest Confusion matrix

The below table contains summary of various score evaluated for models

Accuracy

	Training	**Testing**
Decision Tree	**0.969**	**0.972**
Random Forest	**0.983**	**0.993**

F1 Score

	Training	**Testing**
Decision Tree	**0.959**	**0.96**
Random Forest	**0.978**	**0.99**

8. Comparison:

We can see that random forest beats decision tree in all aspects in training and testing. This is expected since decision tree suffers from high variance, whereas random forest being an ensemble method removes high variance . Trees in random forest are quite diverse which helps to capture the real medical data distribution.

The dataset coming from medical domain requires a careful study of false negatives since they misclassify an ill patients which will harmful to the patients, In that regard also, random forest have 0 false negatives which is good sign.

Covid Analysis

Problem Statement:

Using the LASSO Regression, RIDGE Regression and Principal Component Regression to predict the number of deaths in the Covid Dataset and comparing results.

Data Description:

- ➔ The Covid dataset was allocated to me. The task was to predict the number of deaths given all the other numerical parameters.
- ➔ The dataset has a shape of (230, 11).
- ➔ The shape indicates that we have 10 features, 1 target column and 230 data points.

#	Country, Other	Total Cases	Total Deaths	Total Recovered	Active Cases	Serious, Critical	Tot Cases/ 1M pop	Deaths/ 1M pop	Total Tests	Tests/ 1M pop	Population
1	USA	9,81,66,904	10,84,282	9,49,62,112	21,20,510	2,970	2,93,206	3,239	1,11,81,58,870	33,39,729	33,48,05,269
2	India	4,45,87,307	5,28,629	4,40,19,095	39,583	698	31,698	376	89,44,16,853	6,35,857	1,40,66,31,776
3	France	3,53,42,950	1,55,078	3,45,27,115	6,60,757	869	5,38,892	2,365	27,14,90,188	41,39,547	6,55,84,518
4	Brazil	3,47,06,757	6,86,027	3,38,38,636	1,82,094	8,318	1,61,162	3,186	6,37,76,166	2,96,146	21,53,53,593
5	Germany	3,33,12,373	1,49,948	3,23,15,200	8,47,225	1,406	3,97,126	1,788	12,23,32,384	14,58,359	8,38,83,596

Table 1: A sample of given dataset

Exploratory Data Analysis:

1. Dataset consists of the following columns:

- `Country, Other`: The Country name (string)
- `Total Cases`: The total number of cases in each country
- `Total Deaths:` The number of deaths in the country to date. (The feature to be predicted)
- `Total Deaths:` The number of recoveries in the country to date.
- `Active Cases:` The current active cases in the country.
- `Serious, Critical:` The current number critical patients in the country.
- `Tot Cases/1M pop`: The current number cases per every 1 Million people.
- `Deaths/1M pop:` The number of deaths per every 1 Million people.
- `Total Tests:` The total number of tests done.
- `Tests/1M pop:` The number of tests per every 1 Million people.
- `Population:` The population of the country.

2. Description of features is obtained using df.describe() after correctly parsing all values as numbers and dropping the country column.

	Total Cases	Total Deaths	Total Recovered	Active Cases	Serious, Critical	Tot Cases/ 1M pop	Deaths/ 1M pop	Total Tests	Tests/ 1M pop	Population
count	2.300000e+02	2.250000e+02	2.140000e+02	2.150000e+02	147.000000	228.000000	223.000000	2.140000e+02	2.140000e+02	2.280000e+02
mean	2.705969e+06	2.909820e+04	2.807255e+06	5.744895e+04	270.224490	179621.846491	1197.968610	3.137011e+07	2.050888e+06	3.484620e+07
std	8.779899e+06	1.022137e+05	8.815291e+06	2.073823e+05	922.698201	182372.507626	1246.991628	1.144561e+08	3.490517e+06	1.389233e+08
min	9.000000e+00	1.000000e+00	2.000000e+00	0.000000e+00	1.000000	16.000000	2.000000	5.117000e+03	5.091000e+03	7.990000e+02
25%	2.364900e+04	2.020000e+02	1.968325e+04	1.320000e+02	4.500000	16932.500000	157.500000	3.462602e+05	1.807408e+05	5.170965e+05
50%	2.037110e+05	2.179000e+03	2.250110e+05	1.246000e+03	18.000000	117976.000000	777.000000	2.172044e+06	8.717415e+05	5.816378e+06
75%	1.256286e+06	1.412200e+04	1.437441e+06	1.923950e+04	101.000000	293122.750000	1953.000000	1.267585e+07	2.345814e+06	2.254928e+07
max	9.816690e+07	1.084282e+06	9.496211e+07	2.120510e+06	8318.000000	703959.000000	6429.000000	1.118159e+09	2.200494e+07	1.448471e+09

Table 2: Data description for numerical features

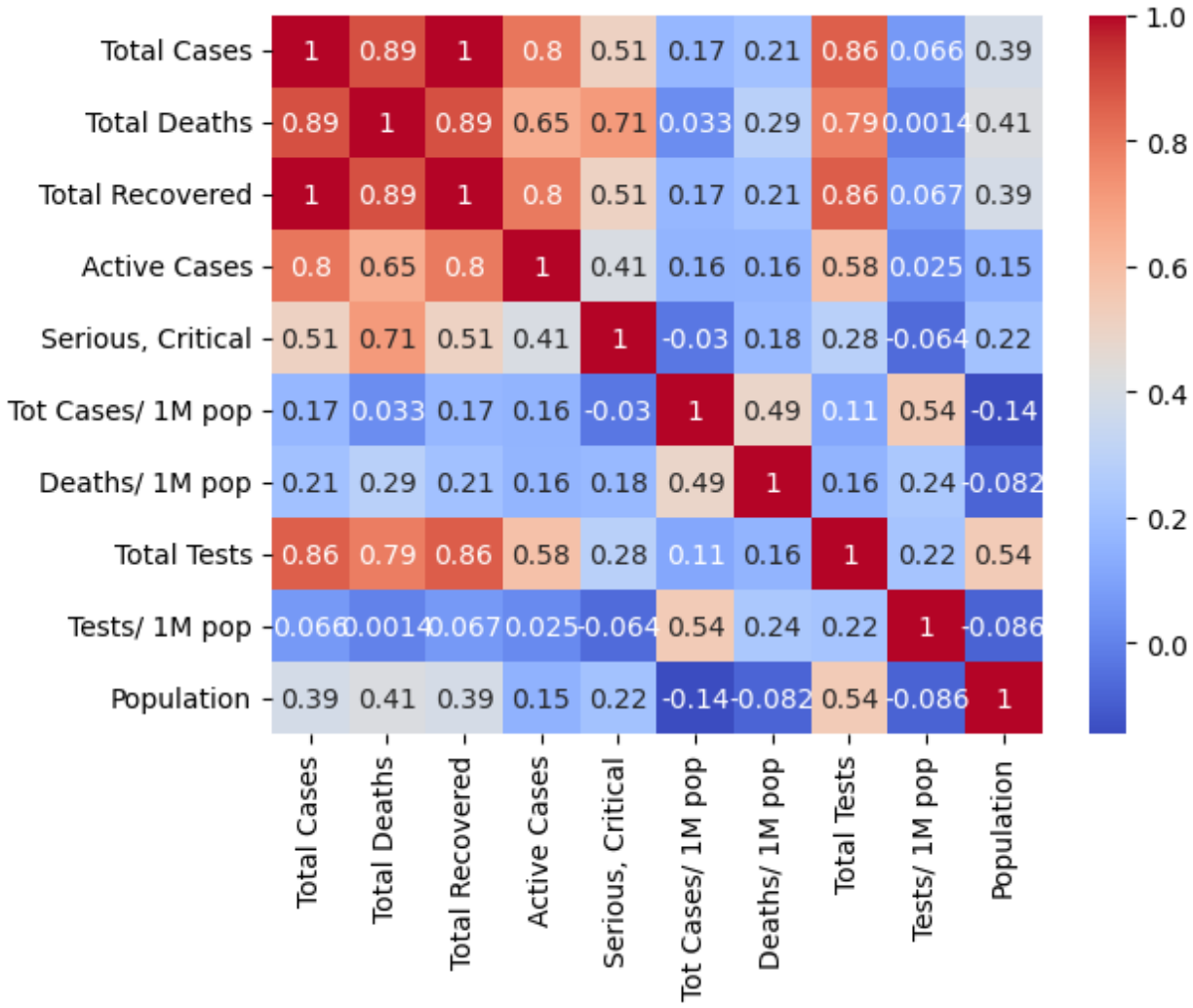
3. There are several null values in the dataset

Several ‘Serious, Critical’ values are missing. Removing them would mean removing a lot of data. So the null values ‘Serious, Critical’ are replaced by 0s. The remaining null values are removed.

```
Total Cases           0
Total Deaths          5
Total Recovered      16
Active Cases         15
Serious, Critical    83
Tot Cases/ 1M pop     2
Deaths/ 1M pop        7
Total Tests          16
Tests/ 1M pop        16
Population            2
dtype: int64
```

```
Index: 200 entries, 1 to 224
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Total Cases        200 non-null    int64
 1   Total Deaths       200 non-null    float64
 2   Total Recovered    200 non-null    float64
 3   Active Cases       200 non-null    float64
 4   Serious, Critical  200 non-null    float64
 5   Tot Cases/ 1M pop  200 non-null    float64
 6   Deaths/ 1M pop     200 non-null    float64
 7   Total Tests        200 non-null    float64
 8   Tests/ 1M pop      200 non-null    float64
 9   Population         200 non-null    float64
dtypes: float64(9), int64(1)
memory usage: 17.2 KB
```

4. We have plotted the values of correlation between features using df.corr()



Plot 2: Correlation plot

Due to the low correlation between Total deaths and test/1M pop and Tot Cases/ 1M pop, these columns can be removed from the dataset. Total Deaths/ 1M pop can also be removed as it is just a scaled version of the data to be prediicted.

Regression

1. Train Test Split

1.1. We remove the unnecessary columns as explained in the EDA section.

1.2. We get the X_train by keeping all columns except the total deaths, and the Y_train is the total deaths.

1.3. We split the dataset into a training set and a test set.

1.4. 80% of the data points are in the training set, and the remaining 20% are in the test.

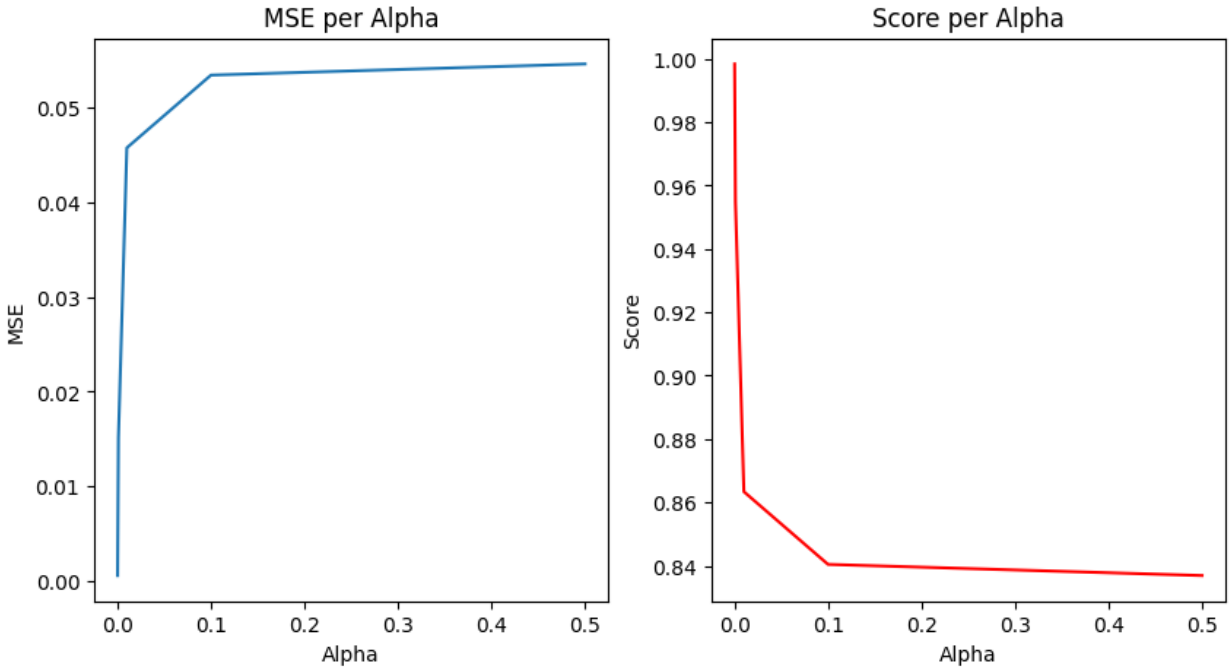
2. Pre processing Data

2.1. We standardise the data using the StandardScaler() class. This data will now be used for training the Regression models.

3. Ridge Multiple Linear Regression

3.1. We train the training data on the Ridge Model using the sklearn.linear_model.Ridge().

3.2. We then test the Mean Square Error (MSE) and the model score (R^2) for various values of alpha.

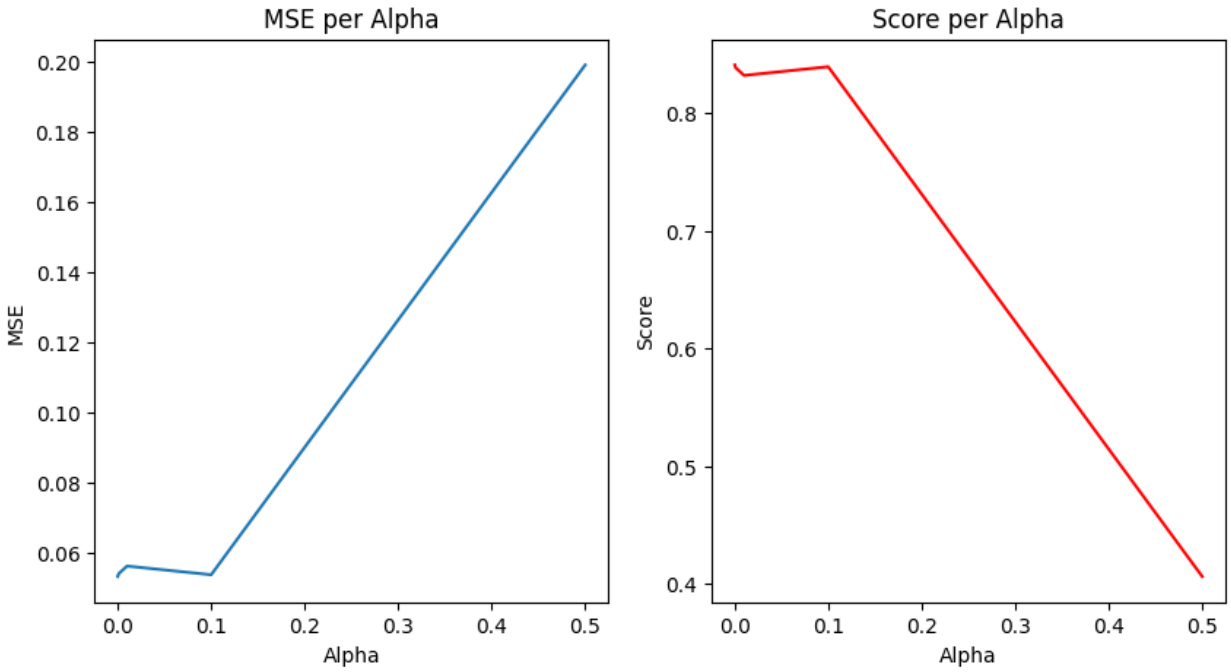


3.3. We get the lowest error and highest score for alpha = 0.0001. The best values of MSE and Model Score are :
Minimum MSE: 0.00054
Maximum Score: 0.99

4. Lasso Multiple Regression

4.1. We train the training data on the Lasso Model using the sklearn.linear_model.Ridge().

4.2. We then test the Mean Square Error (MSE) and the model score (R^2) for various values of values of alpha.



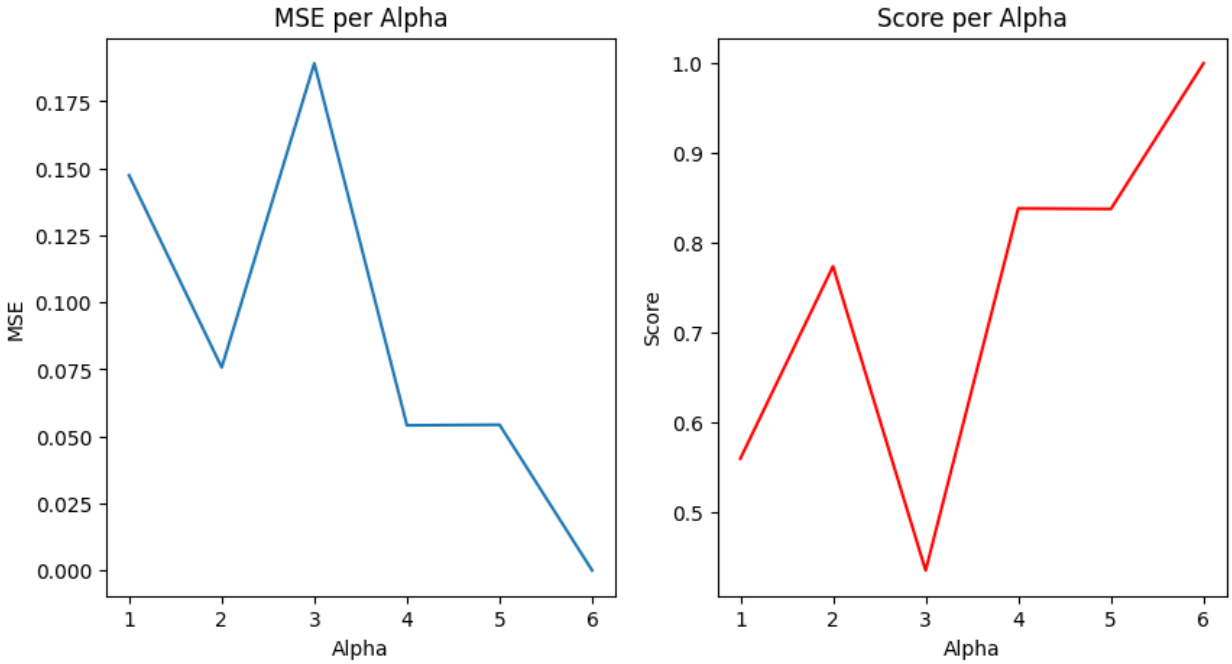
4.3. We get the lowest error and highest score for alpha = 0.0001. The best values for MSE and model score are:
Minimum MSE: 0.0532
Maximum Score: 0.841

5. PCA Multiple Regression

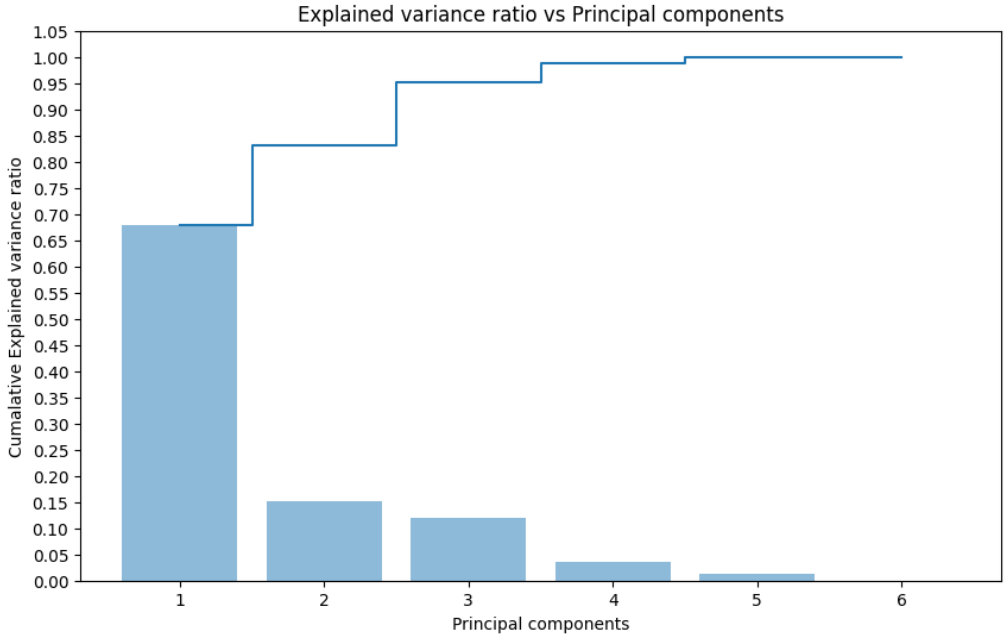
5.1. We transform the data using the sklearn.decomposition.PCA() c,ass

5.2. We train the training data on the simple Linear Regression Model using the sklearn.linear_model.LinearRegression()

5.3. We then test the Mean Square Error (MSE) and the model score (R^2) for various values of the number of components chosen.



5.4. We get the lowest error and highest score for the number of components = 6. The best values for MSE and Model Score are:
Minimum MSE: 8.681e-28
Maximum Score: 1.0



Comparing Results

We compare the results of the three methods in this table:

Model	**Score**	**Mean Square Error**
Ridge Model	0.000549	0.998
Lasso Model	0.053223	0.841
PCA Model	$8.681*10^{-28}$	1.0

This shows that PCA is the best model that provides the least error and maximum score. The Ridge Model follows this, and then the Lasso Model.

PCA > Ridge > Lasso

Stroke Analysis

1. Problem Statement

Use logistic regression and KNN with different K to predict whether a person will suffer a heart stroke. Make the AUC and ROC curves. Compare the findings from different methods. Do exploratory data analysis on the data.

2. Data Description

- Our objective is to perform classification on whether a person will suffer a heart stroke or not, based on the available parameters.
- The shape of the dataset is (5110, 12)
- The shape indicates that we have 11 features, 1 target column and 5110 data points.

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
5	56669	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
6	53882	Male	74.0	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
7	10434	Female	69.0	0	0	No	Private	Urban	94.39	22.8	never smoked	1
8	27419	Female	59.0	0	0	Yes	Private	Rural	76.15	NaN	Unknown	1
9	60491	Female	78.0	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1

Table-1: Sample from the given dataset

3. Exploratory Data Analysis

1. Dataset consists of the following columns:

- Id, gender, age, hypertension, heart_disease, ever_married, work_type. Residence_type. avg_glucose_level, bmi, smoking_status, stroke

- The column id has not been considered during the classification process because it has no influence on a person suffering from a stroke.
- All other parameters are likely to play some part in the chances of a person suffering a stroke.

2. Description of features is obtained using df.describe().

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

Table-2: Description of numerical features

3. Description of categorical features using df.describe(include='object')

	gender	ever_married	work_type	Residence_type	smoking_status
count	5110	5110	5110	5110	5110
unique	3	2	5	2	4
top	Female	Yes	Private	Urban	never smoked
freq	2994	3353	2925	2596	1892

Table-3: Description of categorical features

4. Converted all available categorical variables to numerical parameters using the OneHotEncoder.

5. The column BMI contains 201 columns containing null. However, of the total data available, only 249 rows contain stroke value of 1 of which 40 rows contain BMI value of NaN. Hence to avoid dropping almost 20% of positive data, we have set BMI to the average of all other BMI values in the same stroke group.

```
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
```

Table-4: Null entries in various columns

5. We plotted histograms to study the patterns of each feature using df.hist(). Age, avg_glucose_level and bmi seem to have approximately normal features. id is not going to be used during prediction.

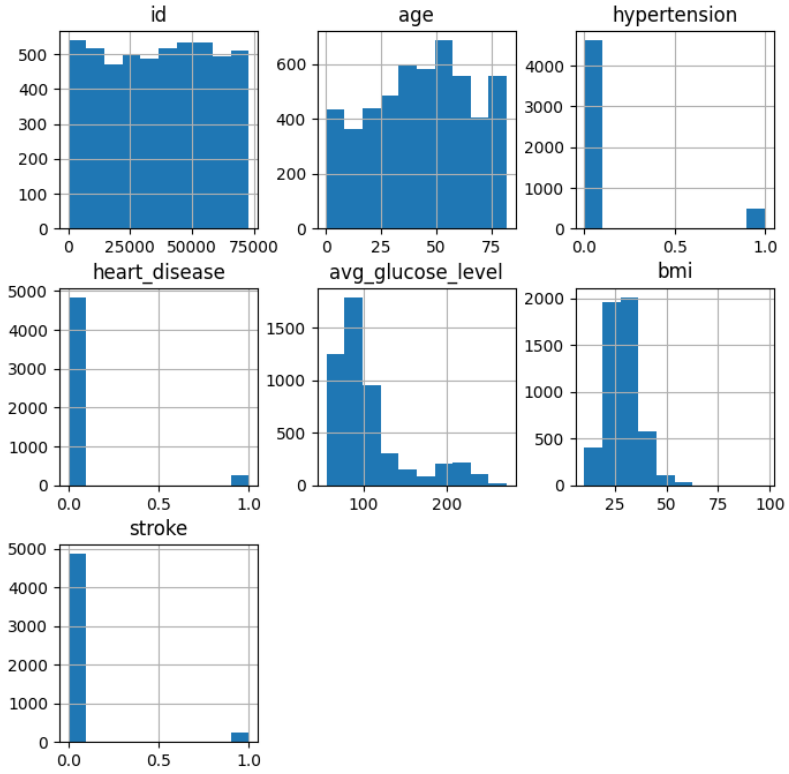


Figure-1: Histogram of feature distribution

6. We have plotted the values of correlation between features using df.corr(). The variables seem to be uncorrelated. All of them can be used during prediction.

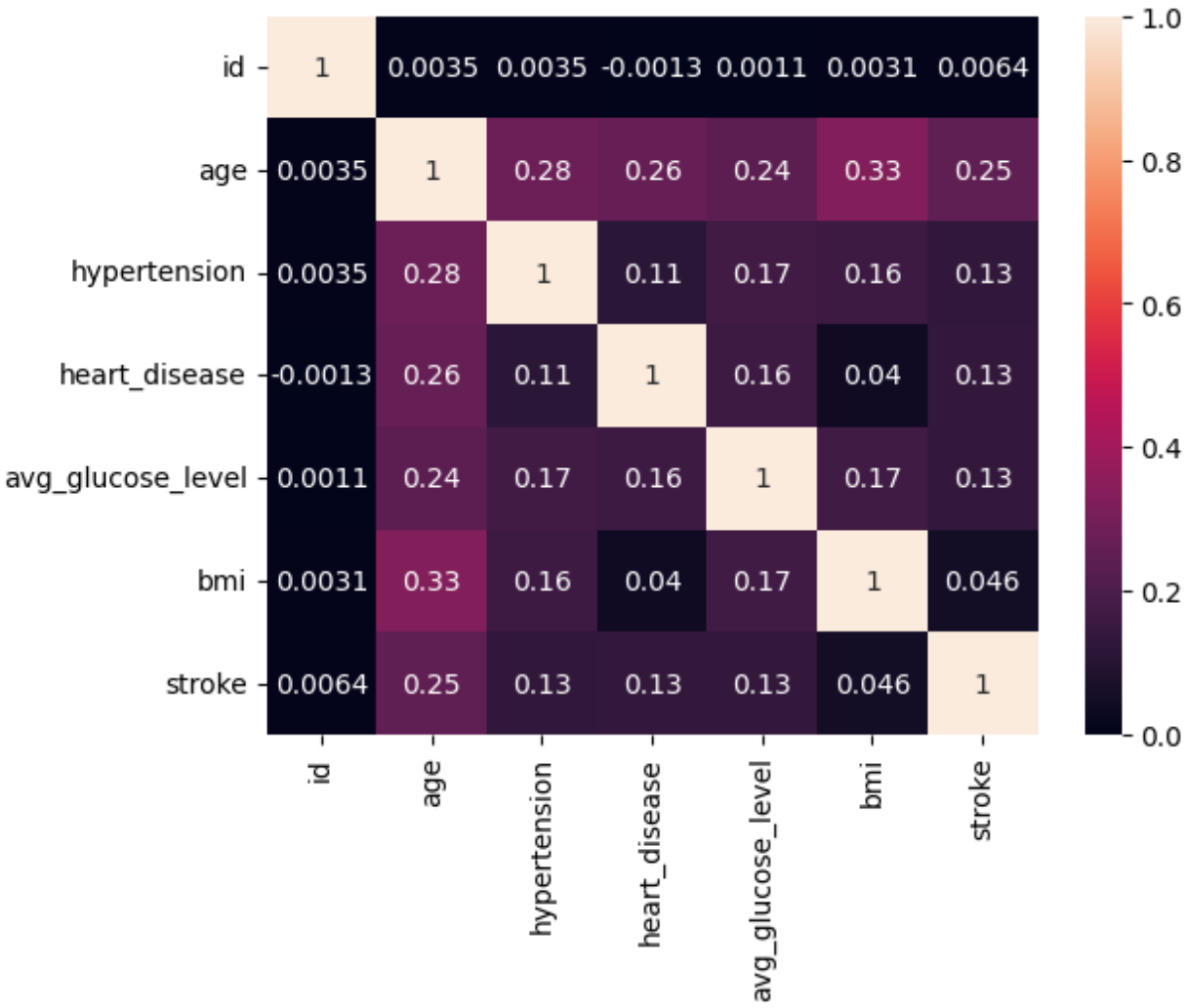


Figure-2: Correlation between dataset parameters

4. Split

We split the data into 80% training data and 20% test data. As the dataset is large, this gives ample number of points for both testing and training. The same split is used across both logistic regression and KNN.

5. Logistic Regression

- We trained the dataset on the training data and tested the classifier on the training dataset. The accuracy using a threshold of 0.5 was 95.3%. Data was standardised using StandardScaler before training.

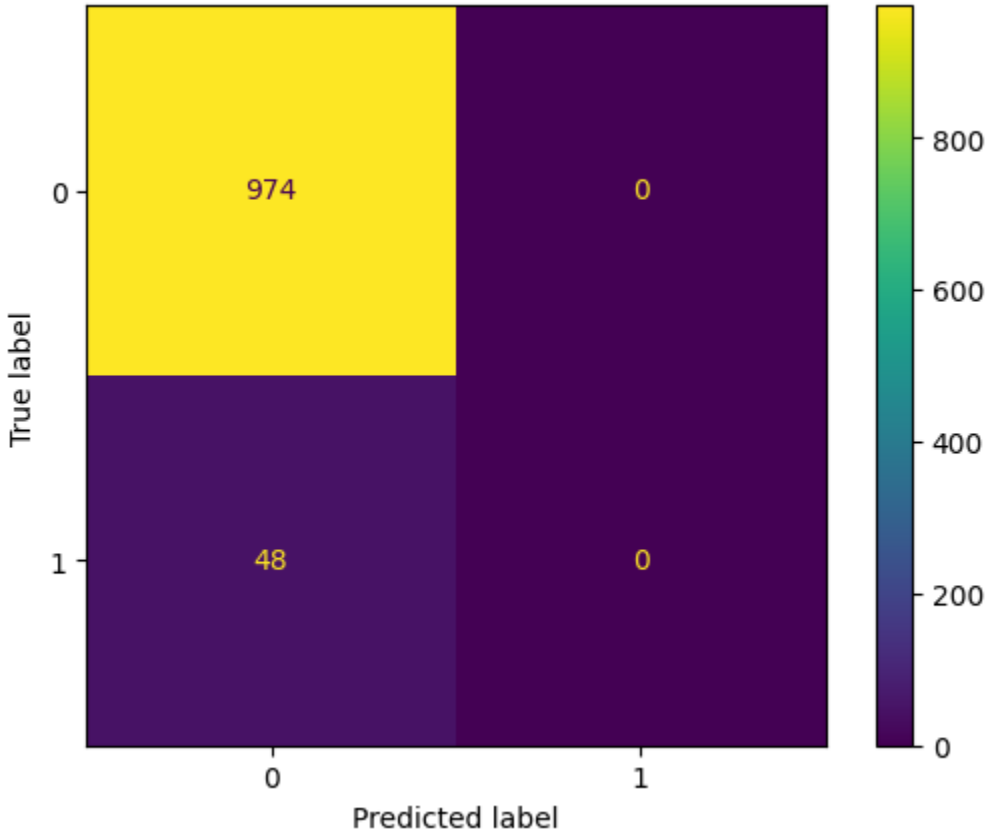


Figure-3: Confusion matrix for logistic regression when threshold = 0.5

- However, because the dataset contains very few points where the stroke is 1, the classifier does not label any of the points as 1 using the default threshold of 0.5. To avoid this, we can change the threshold to improve the TPR.

```
Threshold: 0.10
Test accuracy: 84.83%, TPR: 58.33%, FPR: 13.86%
Confusion Matrix
[[839 135]
 [ 20  28]]

Threshold: 0.20
Test accuracy: 92.27%, TPR: 18.75%, FPR: 4.11%
Confusion Matrix
[[934  40]
 [ 39   9]]

Threshold: 0.30
Test accuracy: 94.42%, TPR: 10.42%, FPR: 1.44%
Confusion Matrix
[[960  14]
 [ 43   5]]

Threshold: 0.40
Test accuracy: 95.30%, TPR: 2.08%, FPR: 0.10%
Confusion Matrix
[[973   1]
 [ 47   1]]

Threshold: 0.50
Test accuracy: 95.30%, TPR: 0.00%, FPR: 0.00%
Confusion Matrix
[[974   0]
 [ 48   0]]
```

Figure-4: Logistic Regression at different thresholds

- Also considered the optimal threshold using both Youden J statistic (maximal value of tpr – fpr) and the distance to optimal (0, 1). The threshold came out to be 0.03909.

```
Best threshold: 0.03909
Test accuracy: 72.60%,
TPR: 83.33%,
FPR: 27.93%

Confusion Matrix
 [[702 272]
 [  8  40]]
```

Figure-5: Best threshold found using Youden J statistic

- The ROC curve has been plotted. The AUC is 0.8096 which implies that the classification method is significantly better than random assignment to classes.

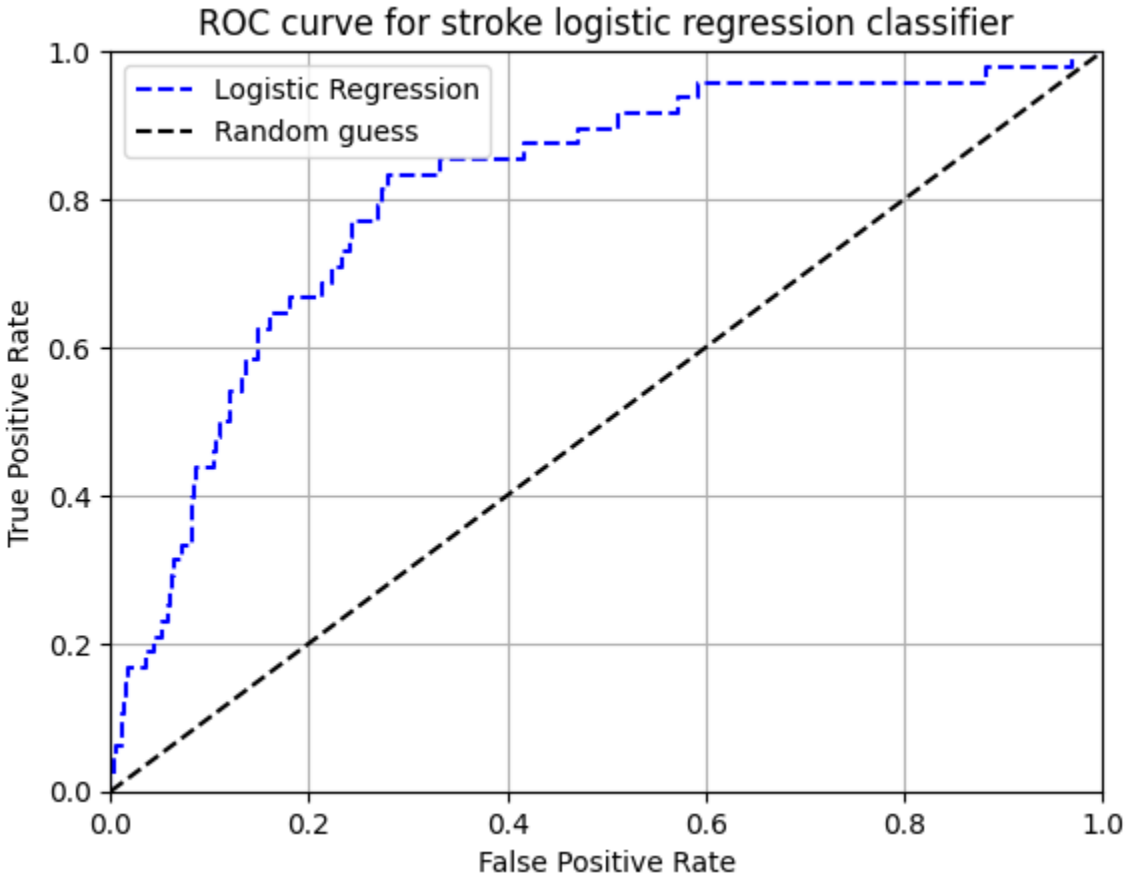


Figure-6: ROC Curve for Logistic Regression

6. KNN

- We trained the KNN classifier on the training dataset using various K values. As the number of data points where stroke is 1 is only 249, using K values above 500 would cause all predictions to be 1. The number of neighbors considered are 10, 25, 50, 100, 200, 250, 300, 400 and 500. Data was standardised using StandardScaler before training.

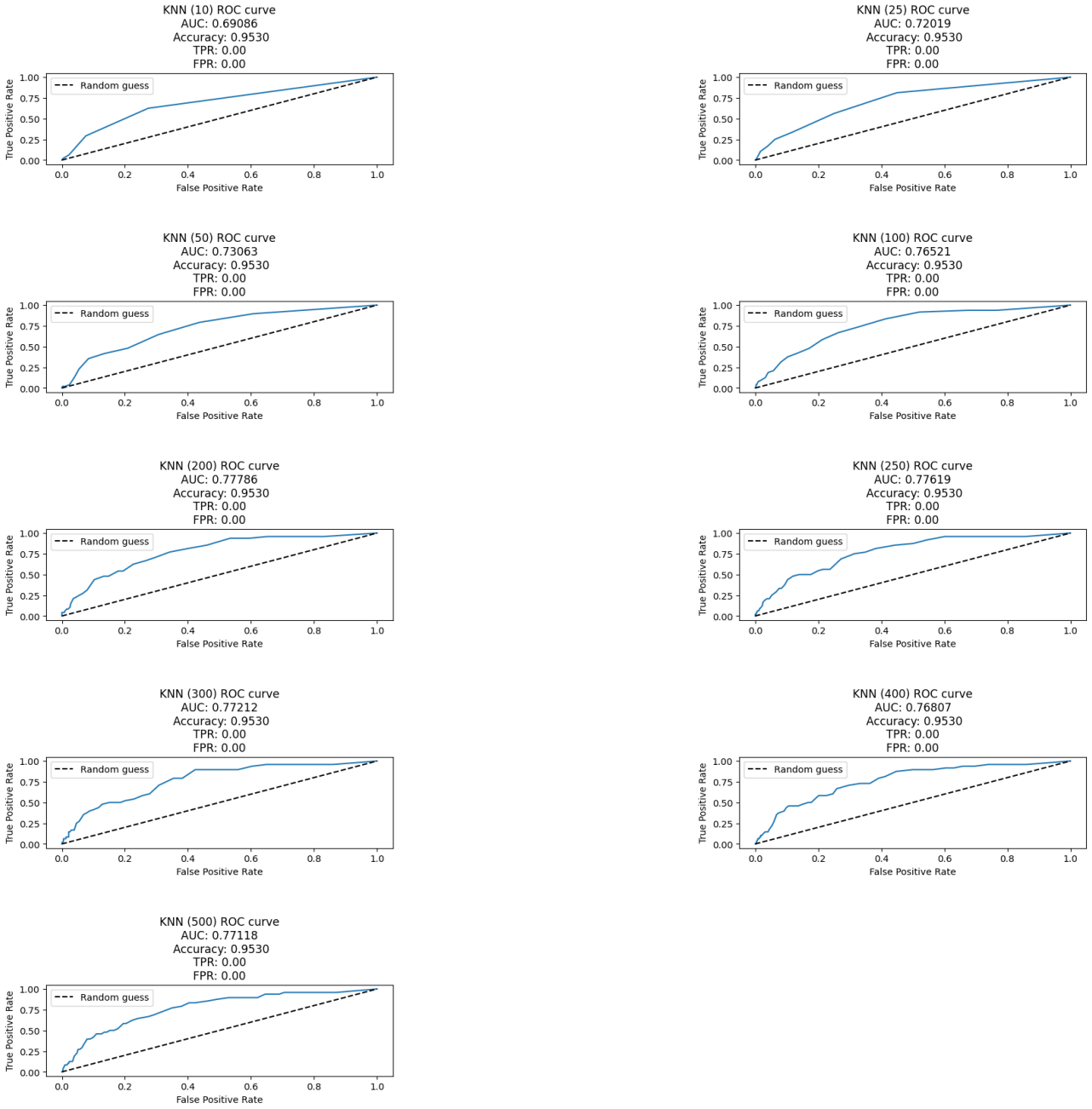


Figure-7: ROC Curve for KNN at various values of K

- Also changing the proportion of neighbors who had to be 1, in order for a point to be classified as 1 with K=50 led to the following results.

```
Threshold: 0.10
Test accuracy: 85.91%, TPR: 34.55%, FPR: 11.17%
Confusion Matrix
[[859 108]
 [ 36  19]]

Threshold: 0.20
Test accuracy: 93.44%, TPR: 3.64%, FPR: 1.45%
Confusion Matrix
[[953  14]
 [ 53   2]]

Threshold: 0.30
Test accuracy: 94.52%, TPR: 0.00%, FPR: 0.10%
Confusion Matrix
[[966   1]
 [ 55   0]]

Threshold: 0.40
Test accuracy: 94.62%, TPR: 0.00%, FPR: 0.00%
Confusion Matrix
[[967   0]
 [ 55   0]]

Threshold: 0.50
Test accuracy: 94.62%, TPR: 0.00%, FPR: 0.00%
Confusion Matrix
[[967   0]
 [ 55   0]]
```

Figure-8: Results at Different Thresholds at K=50

- The optimal threshold which is the proportion of people who had to be 1 to classify as 1 in the case of KNN with 50 neighbors was 0.03846.

```
Best threshold: 0.03846
Test accuracy: 69.47%,
TPR: 94.55%,
FPR: 31.95%

Confusion Matrix
 [[658 309]
 [  3  52]]
```

Figure-9: Optimal Threshold for K=50

7. Comparison of Logistic Regression vs KNN

- Due to the skewed nature of the data towards stroke value of 0, both logistic regression and KNN have difficulties in predicting stroke value of 0.
- However, when considering various thresholds, we notice that logistic regression has a higher AUC and hence acts as a better classifier than KNN for the various values of K that were considered.
- Even when considering the optimal value of threshold, we notice that logistic regression contains a higher accuracy. This implies that using a logistic classifier with a lower threshold might be the way to go because the most important application in this particular case is to predict those patients who have a higher likelihood of suffering strokes.
- As the number of data points increases and the proportion of stroke value being 1 decrease further, KNN will have a harder time finding sufficient number of neighboring points to classify any point into the stroke = 1 class.