Analysis of Cancer Dataset

Regression Method

Team-12

D. Sai Rizwana
S.R. Bhargavi
Shubham Singh
Rohit Menon
Sai Mohan

Bhavan's Vivekananda College | BSC Hons Data Science

Machine Learning Analysis Timeline

1. DATA PRE-PROCESSING
2. EXPLORATORY DATA ANALYSIS
3. TRAINING THE MODEL
4. DETERMINING MODEL ACCURACY
5. COMPARING RESULTS WITH OTHER MODELS
6. CONCLUSION AND INSIGHTS

About The Data

The data was aggregated from a number of sources including the American Community Survey. The task here is to build a multivariate ordinary least square regression model to predict "**TARGET_deathRate**".

Objective

One of the reasons for human death is Cancer. These changes can have many possible causes. Lifestyle habits, genes that you get from your parents, and being exposed to cancer-causing agents in the environment, many times, there is no obvious cause. So the objective is to come up with better analysis and get solutions.

The Path

Implementing multiple machine learning models to fit the best model for the dataset.

Data And Data Quality Check

- ## Data Introduction :

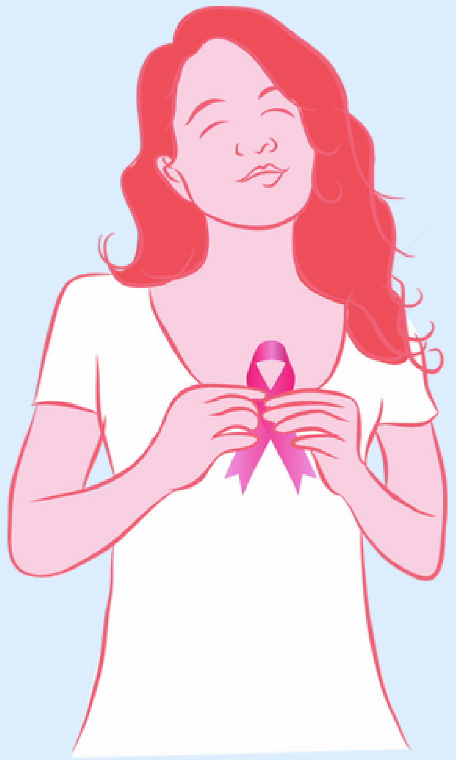
The data consists of 34 columns and 3047 observations from the year 2010-2016 with 2013 census estimates.

- ## Variables :

TARGET_deathRate	**Dependent variable, mean per capita (100,000) cancer mortalities**
medianIncome	Median income per county
povertyPercent	Percent of populace in poverty
PctEmployed16_Over	Percent of county residence ages 16 and over employed
PctUnemployed16_Over	Percent of county residence ages 16 and over unemployed
PctPublicCoverage	Percent of county residence with government provided health coverage
PctPrivateCoverage	Percent of county residence with private health coverage
PctPublicCoverageAlone	Percent of county residence with government provided health coveragealone
PctPrivateCoverageAlone	Percent of county residence with private health coveragealone.

Missing Values :

There were 2 Categorical variables and 32 Continuous variables. There were 3 Continuous variable columns that contained missing values, which were replaced by the mean of the variables, and outliers were identified in the Continuous variable columns.

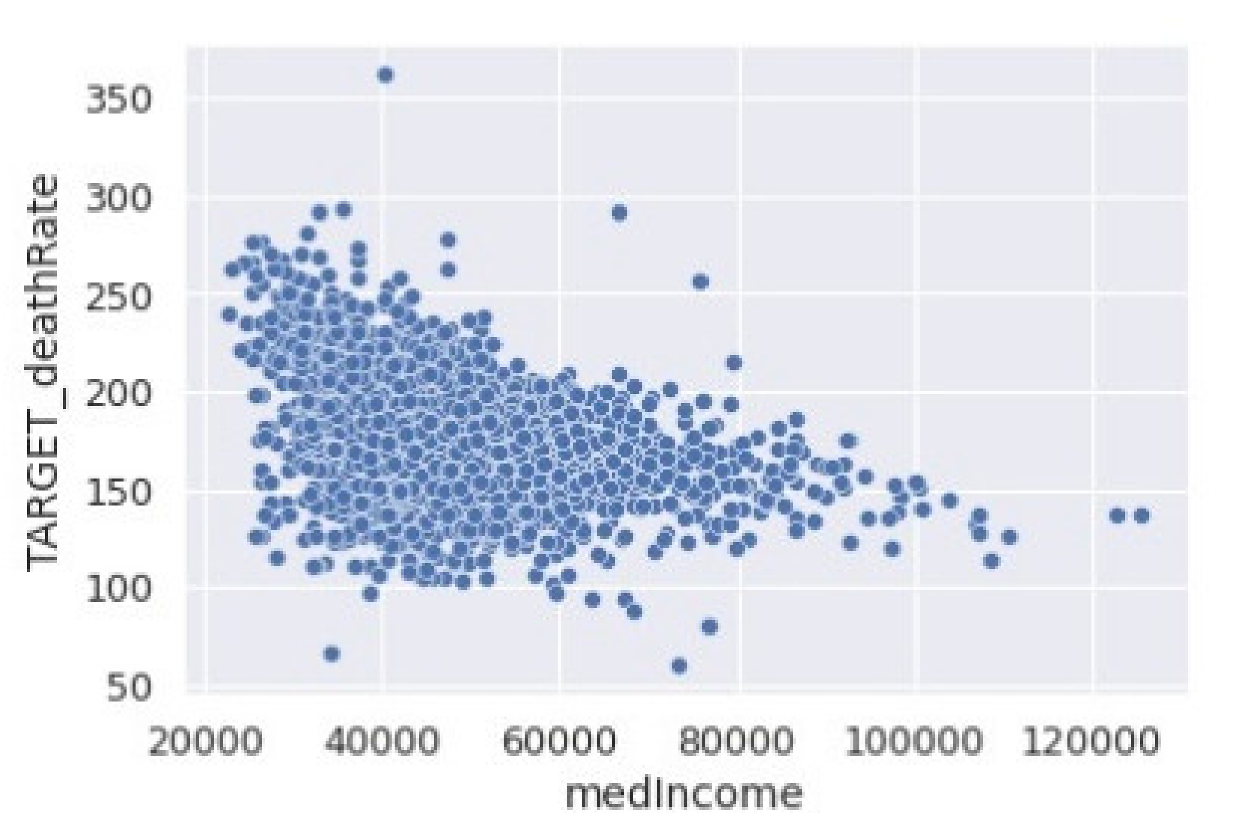


Dropped Columns :

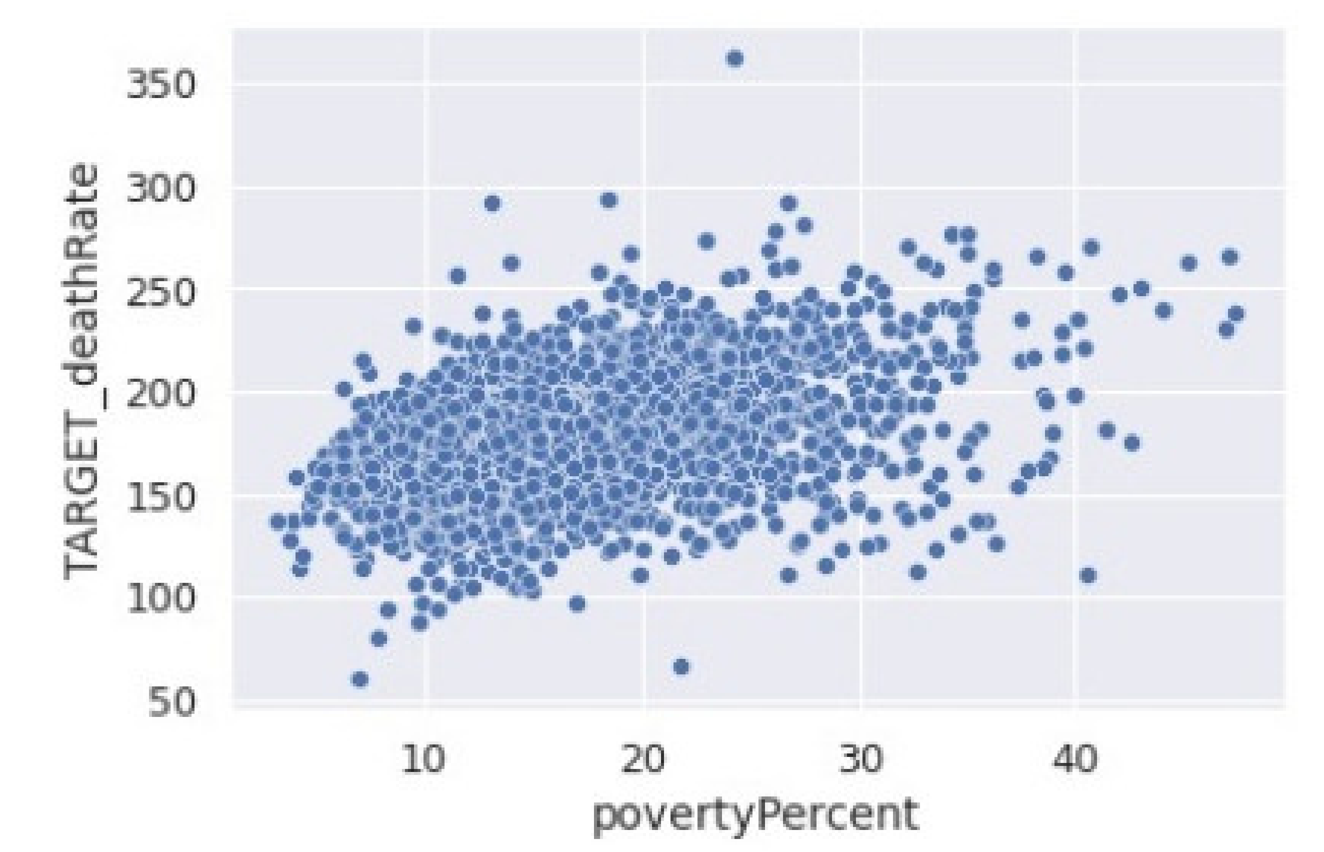
- The Geography column is dropped as it consists of only one type of data in each row.
- studyPerCap column is dropped as it contains many null(0) values.
- The binnedInc column is dropped as we already have medianIncome.
- MedianAgeMale and MedianAgeFemale columns are dropped as we have total MedianAge.
- The PercentMarried column is dropped as we have PctMarriedHouseholds.

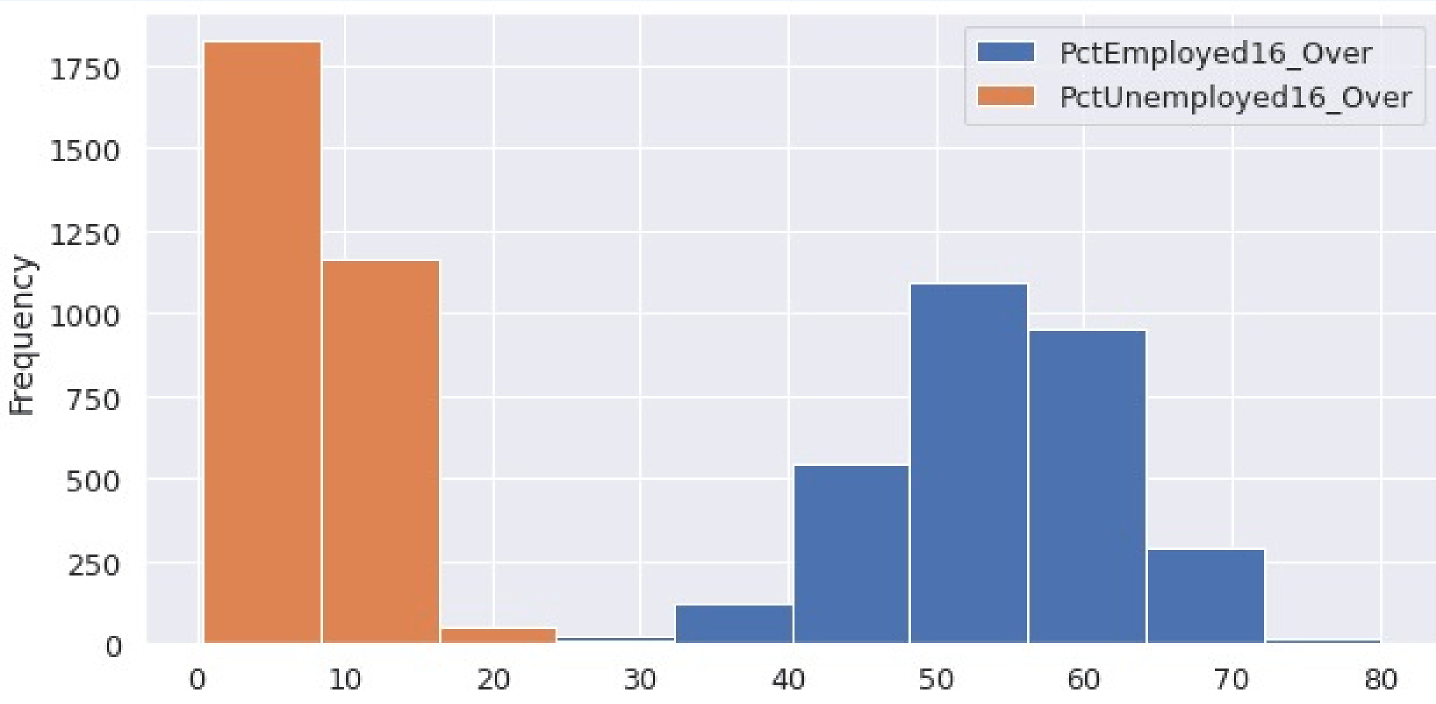
Data Visualization :

- Scatter plot of medIncome shows a high negative correlation with TARGET_deathRate

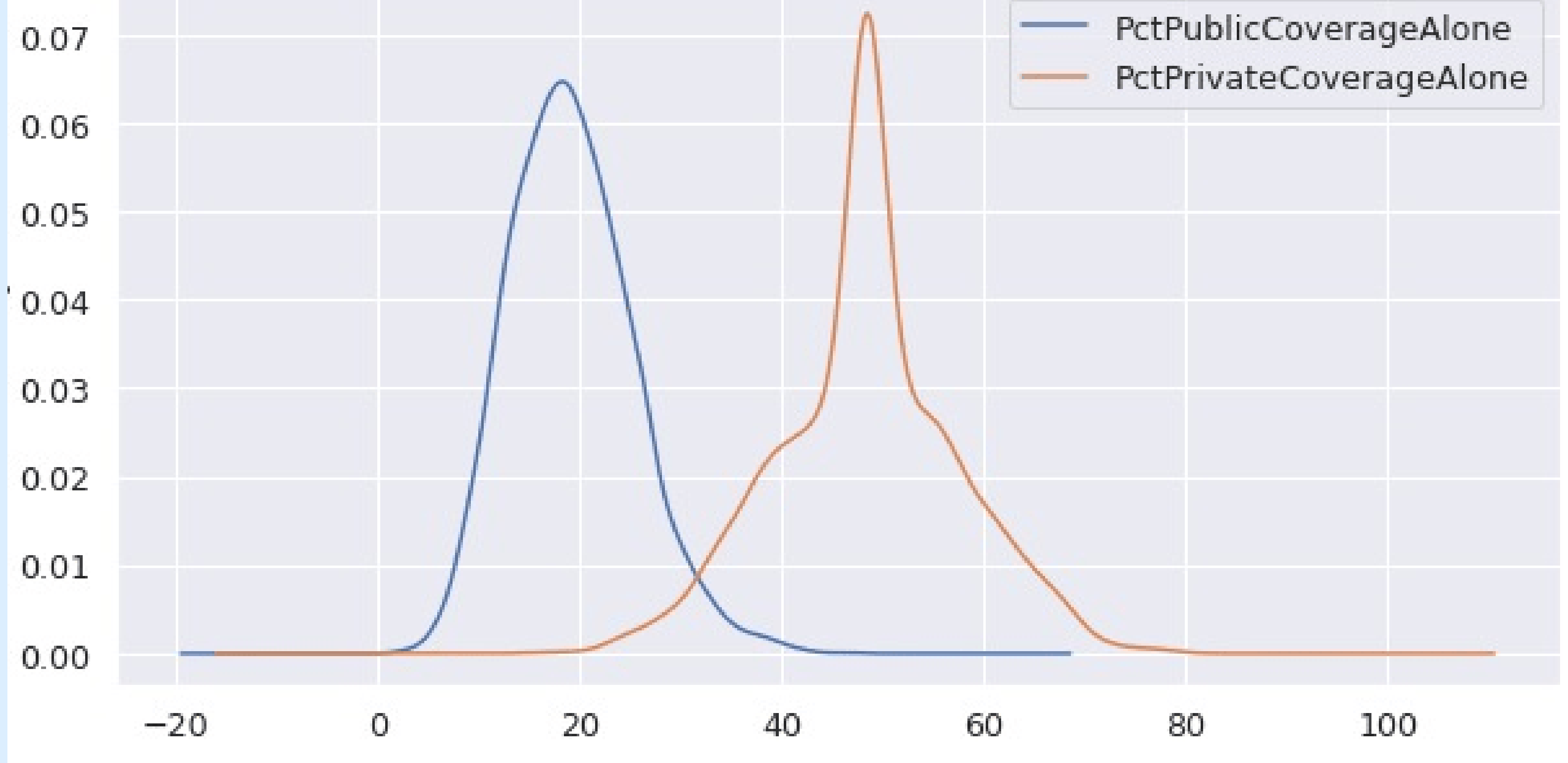


- Scatter plot of povertyPercent shows a high positive correlation with TARGET_deathRate

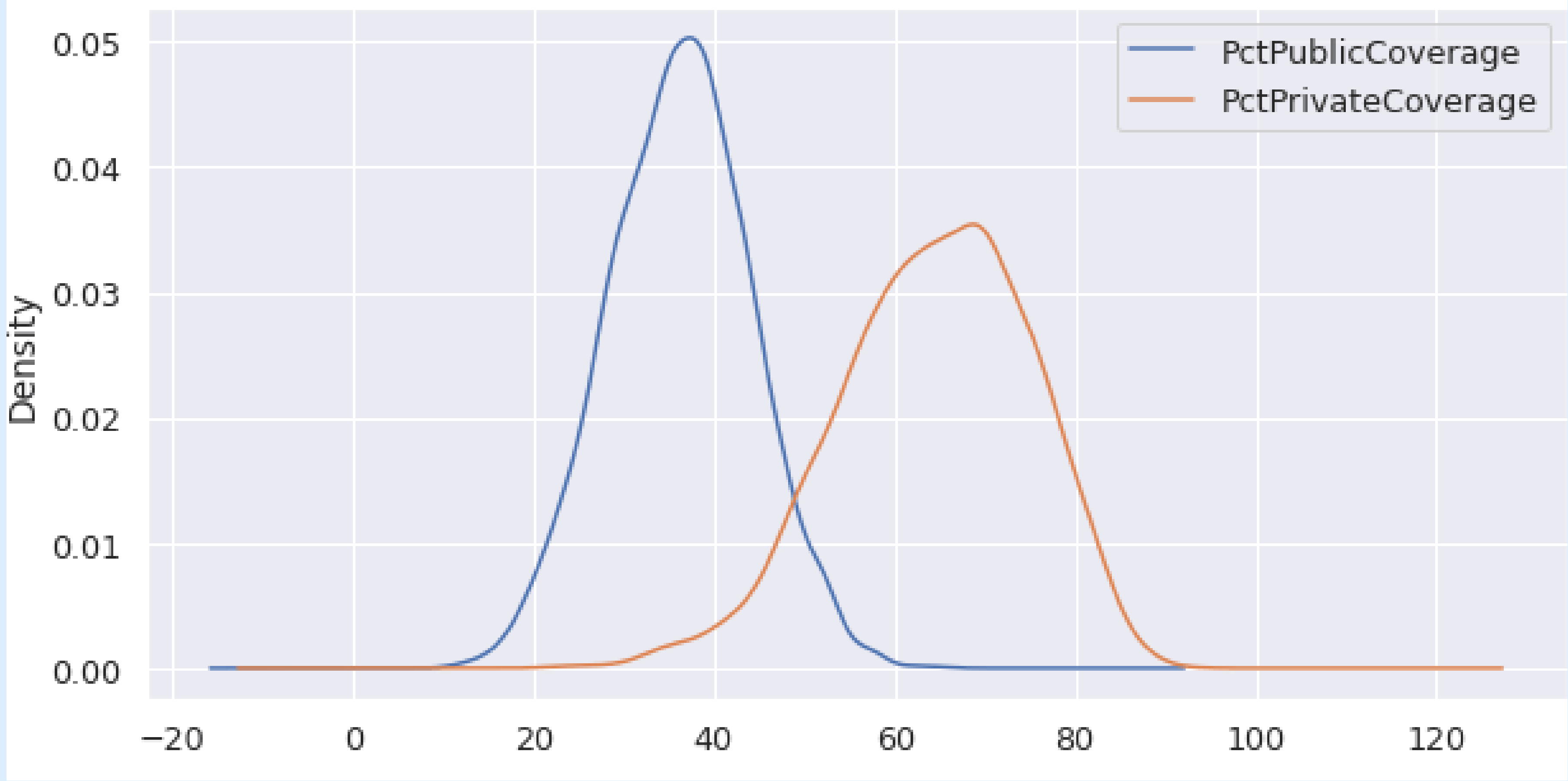




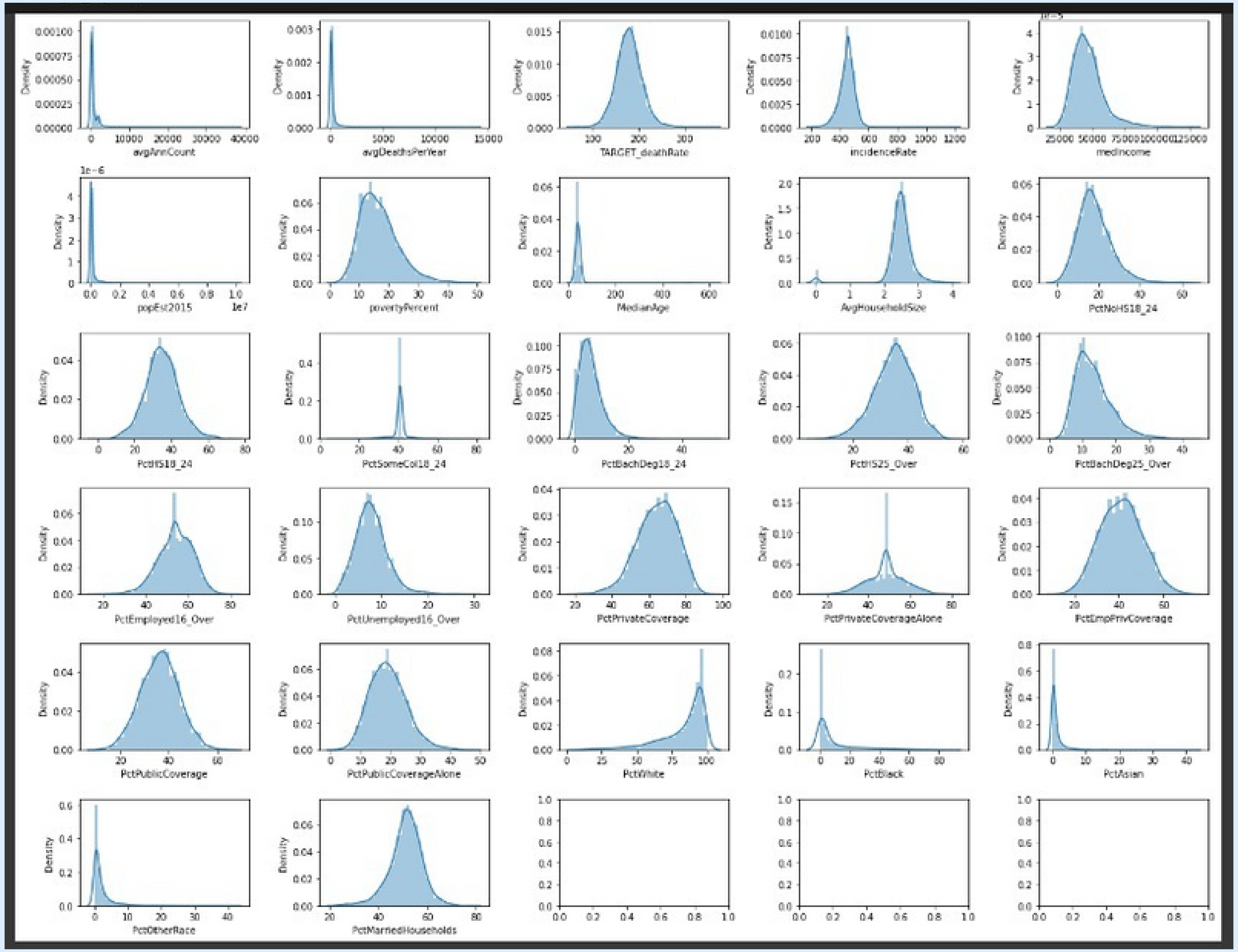
- Hist plot shows how employment percent affects the TARGET_deathRate



- KDE plot shows which type of Alone coverages has impact on the TARGET_deathRate

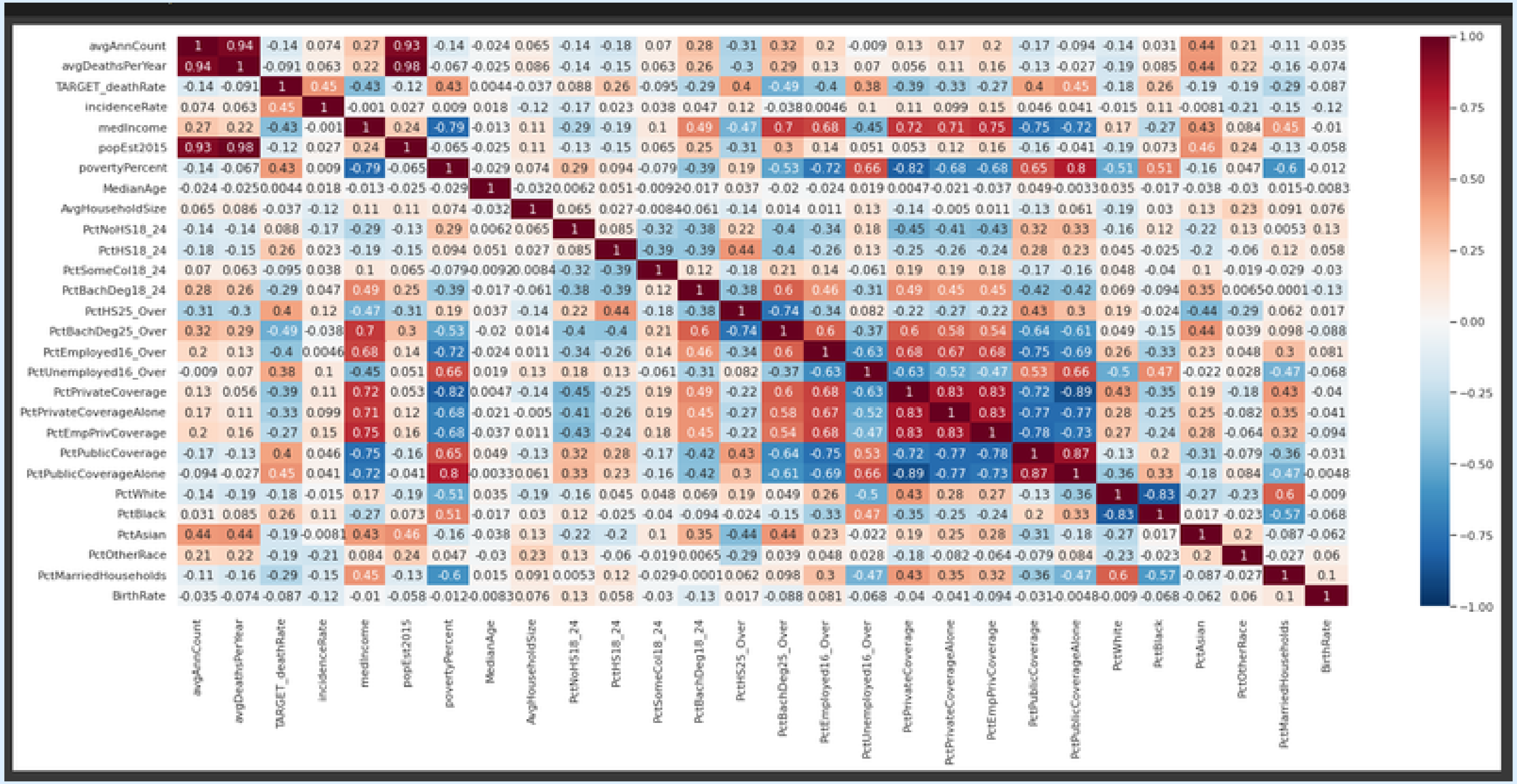


- KDE plot shows which type of family coverages has impact on the TARGET_deathRate



- Most of the features followed the normal distribution.
- PctWhite feature is left skewed.
- PctBlack, PctAsian, PctOtherRace, avgAnncount, avgDeathsPerYear, PopEst2015, MedianAge are right skewed.

Heatmap

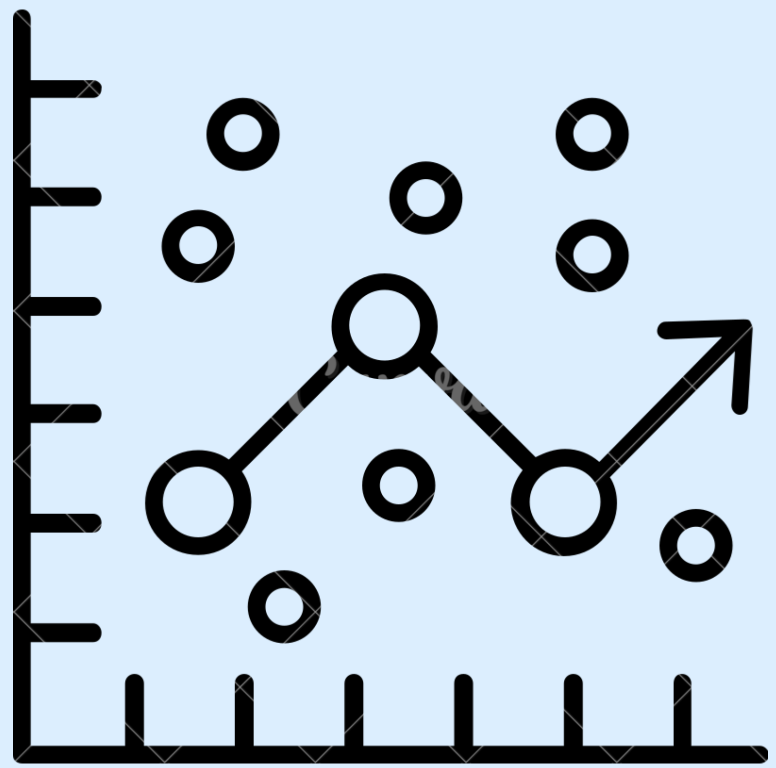


- Incident rate, povertyPercent, PctHs25_Over, PctHs18_24, PctUnemployed16_Over, PctPubliCoverage, PctPublicCoverageAlone, PctBlack are highly positively correlated with target variable(TARGET_deathRate).
- medianIncome, PctBatchDeg25_Over, PctEmpPrivCoverage, PctPrivateCoverage, PctPrivateCoverageAlone, PctEmployed16_Over, PctMarriedHousholds, PctBatchDeg18_24,PctWhite, PctAsian, PctOtherRace are highly negatively correlated with the target variable(TARGET_deathRate)

Algorithms

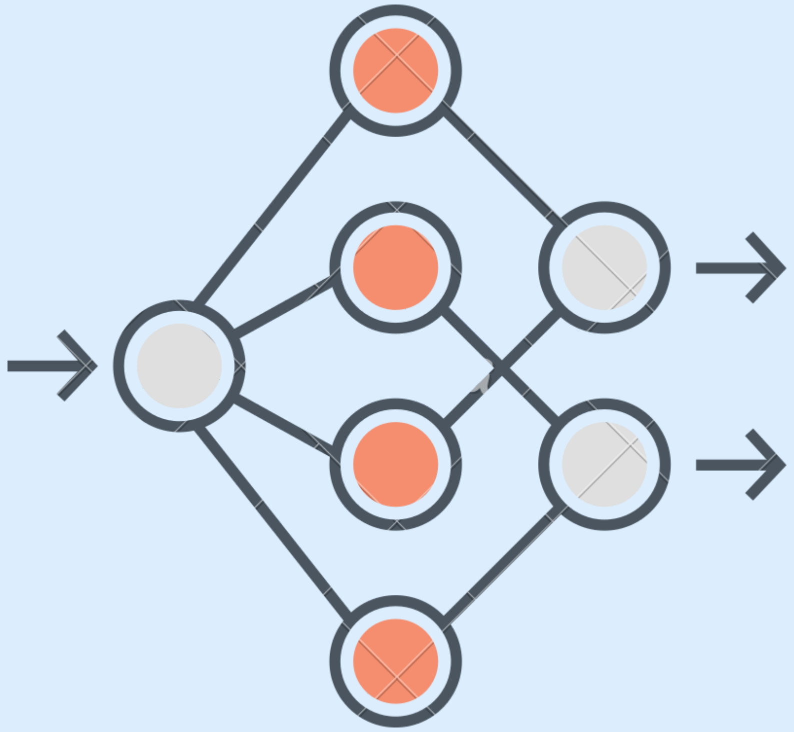
Linear Regression

Train test ratio	R sq value	MAE
65-35	0.46739078	15.39
70-30	0.45	15.75
75-25	0.47	15.22
80-20	0.4779	**15.216**



Appendix Slide No: II

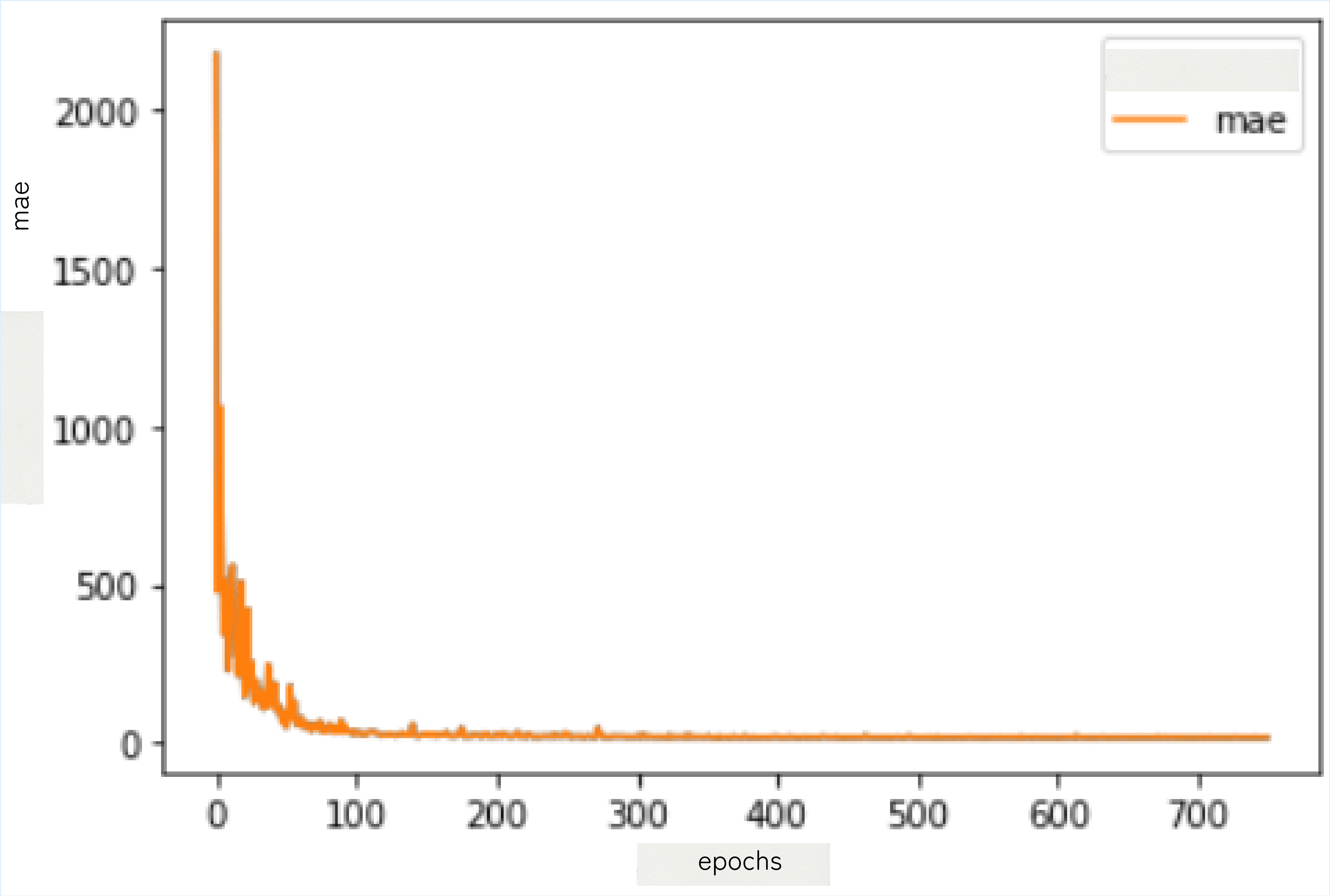
Neural Network



Train-Test	Architecture	Optimizer	Epochs	MAE
65-35	27-20-21-22-4-1	SGD	395	nan
65-35	27-20-21-22-4-1	Adam	395	15.9567
65-35	27-19-12-11-7-1	Adam	550	15.9673
65-35	27-15-15-9-9-1	Adam	1000	16.0941
65-35	27-15-10-5-1	Adam	400	16.1617
70-30	27-300-100-55-20-18-10-6-3-2-1	Adam	750	15.9291
70-30	27-20-12-7-3	Adam	750	16.0296
70-30	27-55-30-17-10-6-3-2-1	Adam	750	16.091
70-30	27-100-71-38-24-16-10-8-5-3-2-1	Adam	750	16.1567
75-25	27-26-13-6-3-1	Adam	1200	16.2356
75-25	27-26-13-6-3-1	Adam	200	17.31231
75-25	27-26-13-6-3-1	Adam	250	18.3123
75-25	27-26-13-7-3-1	Adam	1000	16.5673
80-20	27-13-6-4-1	Adam	750	15.5412
80-20	27-10-9-8-5-1	Adam	200	15.5748
80-20	27-24-12-6-3-1	Adam	750	**15.302**
80-20	27-26-13-7-1	Adam	1000	15.4089

Appendix Slide No: III

Neural Network plot for Observation



train test split	80-20
Architecture	27-24-12-6-3-1
Optimizer	Adam
epochs	750

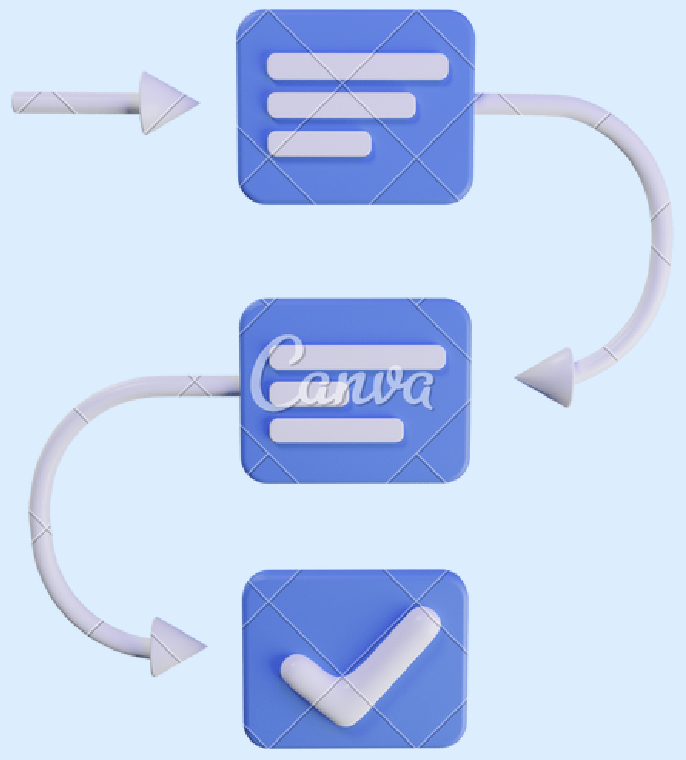
Bagging :

Train Test	n_estimators	MAE
65-35	1000	14.3155
70-30	1000	14.38
75-25	1000	**13.98**
80-20	1000	14.021

Boosting :

Adaboost :

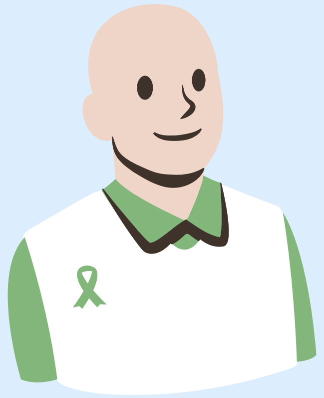
Train Test	learning rate	n_estimators	MAE
65-35	0.9	2000	**15.87**
65-35	0.1	1000	15.96
65-35	0.1	2000	15.97
70-30	0.01	1000	16.34
70-30	0.01	2000	16.44
70-30	0.3	1000	16.48
75-25	0.9	2000	16.32
80-20	0.1	1000	16.776



Appendix Slide No: V

Boosting :

Gradient Boost :

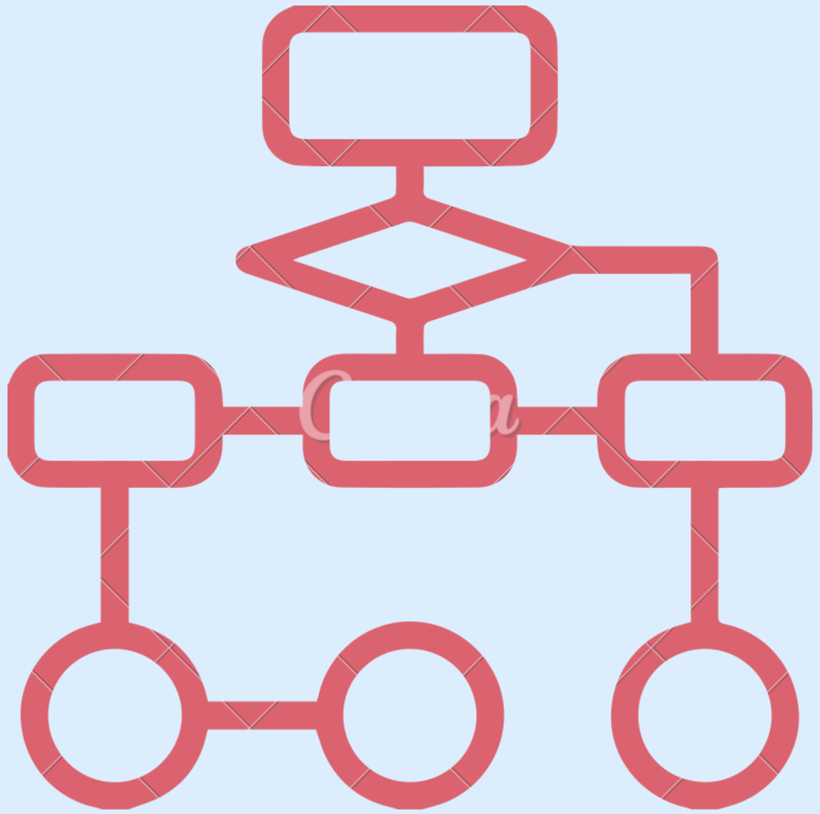


Train Test	learning rate	n_estimators	MAE
65-35	0.1	2000	11.63
65-35	0.1	1000	11.73
65-35	0.3	2000	11.99
70-30	0.1	2000	11.32
70-30	0.1	1000	11.54
70-30	0.3	2000	11.96
75-25	0.1	2000	11.1
80-20	0.1	1000	**10.647**

Boosting :

Extreme Gradient Boosting :

Train Test	n_estimators	learning rate	MAE
65-35	2000	0.1	11.68
65-35	1000	0.1	11.77
65-35	2000	0.3	11.90
70-30	2000	0.1	11.24
70-30	1000	0.1	11.49
70-30	1000	0.3	11.67
75-25	2000	0.1	10.88
80-20	1000	0.1	**10.555**



Appendix Slide No: VII

Comparision of Implemented Models :

Model	Mean Absolute Error
Linear Regression	15.21
Neural Networks	15.30
Bagging	14.02
AdaBoost	16.77
Gradient Boosting	10.64
Extreme Gradient Boosting	**10.55**

Summary And Recommendations

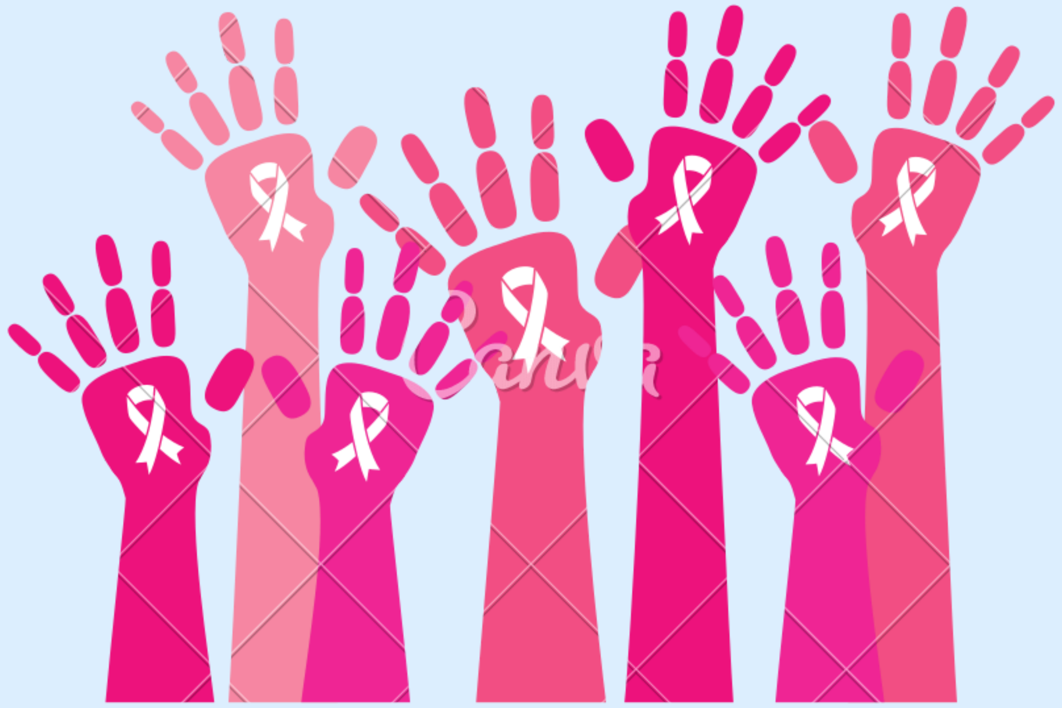
- For the cancer dataset, we performed four types of machine learning algorithms: Linear Regression, Neural Network, Bagging and Boosting.
- The visualization of data performed in exploratory data analysis showed clearly which of the features are more and less correlated with the target variable (TARGET_deathRate).
- From all the analysis and implementations of the algorithm, we found the best results in Boosting algorithm i.e. in the Extreme Gradient Boosting algorithm for 80-20 train-test ratio with Mean Absolute Error (MAE) of 10.555 which is the least error value among all the other errors.
- So, here we can conclude that Extreme Gradient Boosting Algorithm can be considered the best model for the given cancer dataset.

Future insights that contribute to lowering the death rate include the following

- The features like employment percent, income rate, poverty percent, and coverages are interconnected with one another, as we have seen in the data visualization, and this has a significant impact on the target death rate.

- Therefore, to lower the target death rate, it is necessary to raise key variables like the employment rate and income rate. Additionally, having adequate medical facilities and appropriate coverage might aid in lowering a county's death rate.

- Even after accounting for the effects of other variables such as income, level of education was still the basic predictor for death rate due to cancer.



Thank you!

Click here to refer code

Open in Colab

Presented by

D. Sai Rizwana

S.R. Bhargavi

Shubham Singh

Rohit Menon

Sai Mohan

Appendix

Training And Testing :

```
x= data.drop("TARGET_deathRate", axis=1)
y=data["TARGET_deathRate"]
x.head
```

```
[ ]  y.head
```

```
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size=0.2, random_state=42)
y_test
```

```
1582    186.5
2367    152.5
2091    174.2
343     207.6
2661    192.0
        ...
845     213.2
2310    178.0
1412    169.0
2472    161.6
351     199.5
Name: TARGET_deathRate, Length: 610, dtype: float64
```

Linear Regression :

```
#Linear Regression
lm=LinearRegression()
lm.fit(x_train,y_train)
y_pred=lm.predict(x_test)
y_pred
```

```
mae=metrics.mean_absolute_error(y_test,y_pred)
print(mae)
```

```
15.216776762515408
```

```
[ ] r2_score(y_test,y_pred)
```

```
0.4779604905815009
```

Neural Networks :

```
#Imports
%matplotlib inline
import tensorflow as tf
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```
tf.random.set_seed(42)

# STEP1: Creating the model

model= tf.keras.Sequential([
                            tf.keras.layers.Dense(27),
                            tf.keras.layers.Dense(24),
                            tf.keras.layers.Dense(12),
                            tf.keras.layers.Dense(6),
                            tf.keras.layers.Dense(3),
                            tf.keras.layers.Dense(1)
])

# STEP2: Compiling the model   # optimizer can be SGD, Adam

model.compile(loss= tf.keras.losses.mae,
              optimizer= tf.keras.optimizers.Adam(),
              metrics= ["mae"])

# STEP3: Fit the model

history= model.fit(x_train, y_train, epochs= 750, verbose=0)
```

```
model.evaluate(x_test, y_test)
```

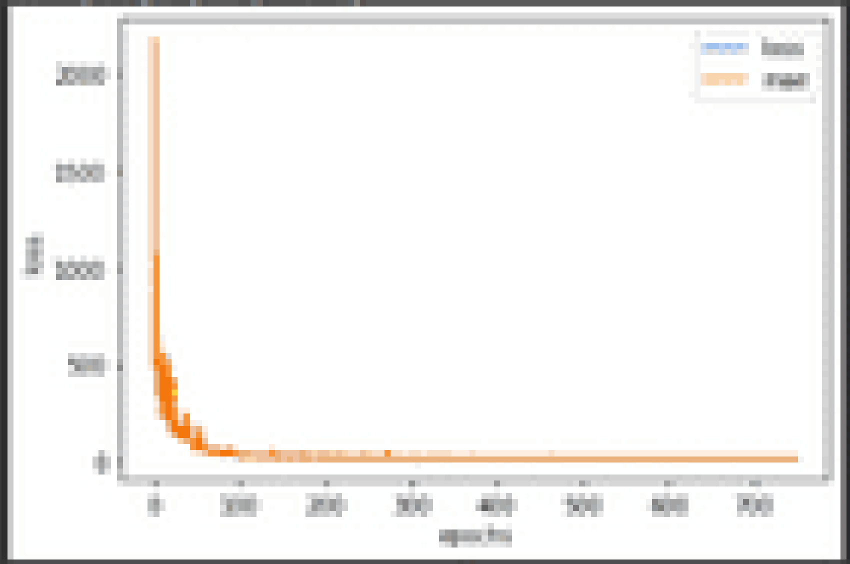
```
20/20 [==============================] - 0s 2ms/step - loss: 15.3020 - mae: 15.3020
[15.301989555358887, 15.301989555358887]
```

```
model.summary();
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 27)                756

 dense_13 (Dense)            (None, 24)                672

 dense_14 (Dense)            (None, 12)                300

 dense_15 (Dense)            (None, 6)                 78

 dense_16 (Dense)            (None, 3)                 21

 dense_17 (Dense)            (None, 1)                 4

=================================================================
Total params: 1,831
Trainable params: 1,831
Non-trainable params: 0
_________________________________________________________________
```

```
pd.DataFrame(history.history).plot()
plt.ylabel("loss")
plt.xlabel("epochs")
```

```
Text(0.5, 0, 'epochs')
```



Bagging :

```
#Bagging
bag_model = BaggingRegressor(
base_estimator=BaggingRegressor(),
n_estimators=1000,
max_samples=0.8,
bootstrap=True,
oob_score=True,
random_state=42
)
```

```
l=bag_model.fit(x_train, y_train)
```

```
mae = metrics.mean_absolute_error(y_test, l.predict(x_test))

print("The mean abs error (MAE) on test set: {:.4f}".format(mae))
```

```
The mean abs error (MAE) on test set: 14.0211
```

Ada Boosting :

```
#Adaptive Boosting
adaclf = AdaBoostRegressor(
                            n_estimators=1000,
                            learning_rate=0.1,
                            random_state=42)


adaclf.fit(x_train, y_train)
y_pred_1 = adaclf.predict(x_test)
ab=mean_absolute_error(y_test, y_pred_1)
print(ab)
```

```
15.960844803603948
```

V

Gradient Boosting :

```
#Gradient Boosting
regressor = GradientBoostingRegressor(
    max_depth=3,
    n_estimators=1000,
    learning_rate=0.1,
    random_state=42
)
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)
gb=mean_absolute_error(y_test, y_pred)
print(gb)
```

```
11.738550600273754
```

VI

Extreme Gradient Boosting :

```
#Extreme Gradient Boosting
clf = XGBRegressor(n_estimators=1000,
                   learning_rate=0.1,
                   max_depth=3,
                   random_state=42)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
eg=mean_absolute_error(y_test, y_pred)
print(eg)
```

```
[12:45:33] WARNING: /workspace/src/objecti
11.775064202749293
```