1. LINEAR REGRESSION MODEL TO PREDICT THE DEATH RATE:

Table of Content:

A) Dataset Description:

This dataset was taken from "data. World" repository,

In this dataset contains 3047 instances and 33 features in these details about mortality rates due to cancer.

I tend to analyse the features and extract them to predict the target variable, and the target features is 'target deathrate'.

Number of features -> 33
Number of observations -> 3047
Number of numerical features: 31
Number of categorical features: 2

Dependent variable -> target_deathrate

List of Numerical Features:

```
In [6]: num_df=data.select_dtypes(include='number')
        num_df.shape
        num_df.columns
```

```
Out[6]: Index(['avganncount', 'avgdeathsperyear', 'target_deathrate', 'incidencerate',
               'medincome', 'popest2015', 'povertypercent', 'studypercap', 'medianage',
               'medianagemale', 'medianagefemale', 'percentmarried', 'pctnohs18_24',
               'pcths18_24', 'pctsomecol18_24', 'pctbachdeg18_24', 'pcths25_over',
               'pctbachdeg25_over', 'pctemployed16_over', 'pctunemployed16_over',
               'pctprivatecoverage', 'pctprivatecoveragealone', 'pctempprivcoverage',
               'pctpubliccoverage', 'pctpubliccoveragealone', 'pctwhite', 'pctblack',
               'pctasian', 'pctotherrace', 'pctmarriedhouseholds', 'birthrate'],
              dtype='object')
```

List of Null Features:

```
In [7]: nan_df=data.loc[:, data.isna().any()]
        nan_df.shape
        nan_df.columns

Out[7]: Index(['pctsomecol18_24', 'pctemployed16_over', 'pctprivatecoveragealone'], dtype='object')
```

List of Categorical Features:

```
In [5]: cat_df=data.select_dtypes(include='object')
        cat_df.head(10)
        cat_df.shape
        cat_df.columns

Out[5]: Index(['binnedinc', 'geography'], dtype='object')
```

Dataset Summary:

- Our target feature deathrate has mean of 178.6 and median of 178.1 which says it's normally distributed.
- 25% death rate are below 161, 75 - 100% death rate is from count more than 195.

B) Treatment Of Missing Values:

- There are 2 ways to treat null values one is univariate imputation in which single columns are used another one is multivariate imputation in which more than 1 column participate in finding missing data.

- Univariate imputation consists mean, median, mode, constant missing indicator and random value imputation also we can drop that row, if there is a column in which percentage of missing data is very high then we can drop that column and if missing data is below 5% then we can use central values to fill that for normally distributed data use either mean, median or mode but for skewed data use median.

- Mean and median of null valued columns are almost same. We cannot remove rows with null values, because 'pctsomecol18_24' is having 2285 null values and removing the rows containing values makes 75% data loss, therefore it heavily affects model prediction.

- Since pctsomecol18_24 has 2285 null values and it doesn't contribute for prediction we can drop it. We can replace null values for pctemployed16_over and pctprivatecoveragealone with their mean values.

- After filling missing values, we should always check for variance, variance should not change much.

- We can see after filling missing values outlier get increased, we can also go with random value imputation it's advantage over central value imputation is that distribution does not change much but the covariance gets distorted with other variables so take care of that.

Pctemployed16_Over:

```
In [12]: data['pctemployed16_over_mean']=data['pctemployed16_over'].fillna(data['pctemployed16_over'].mean())
         data['pctemployed16_over_median']=data['pctemployed16_over'].fillna(data['pctemployed16_over'].median())
         data['pctemployed16_over_mode']=data['pctemployed16_over'].fillna(54)

         print(data['pctemployed16_over_mode'].var())
         print(data['pctemployed16_over_mean'].var())
         print(data['pctemployed16_over_median'].var())
         print(data['pctemployed16_over'].var())

         65.69119572312188
         65.69009103249324
         65.69581165375632
         69.14029622839475
```

- After filling missing values, we should always check for variance, variance should not change much, but in the pctemployed16 over feature column is more variance, (Fig 1)

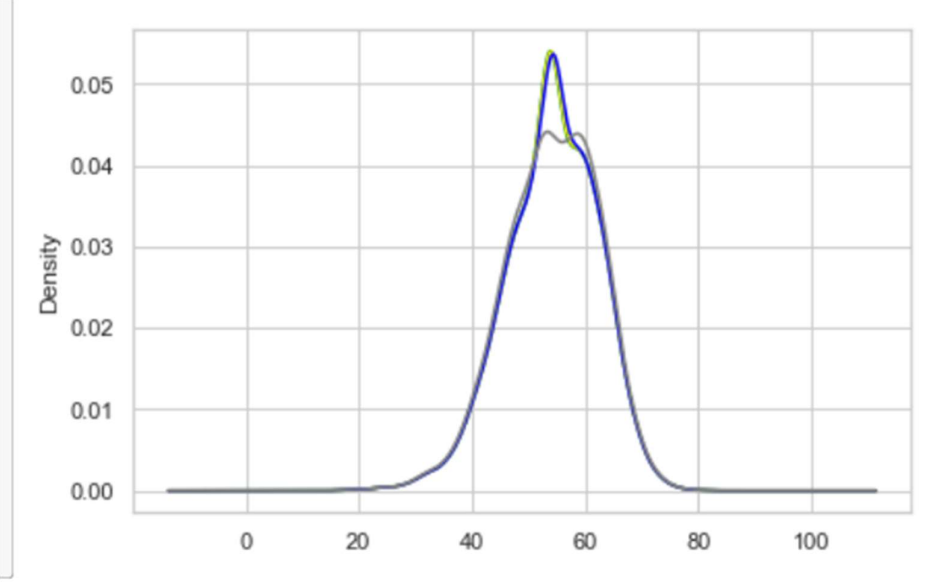


Fig - 1

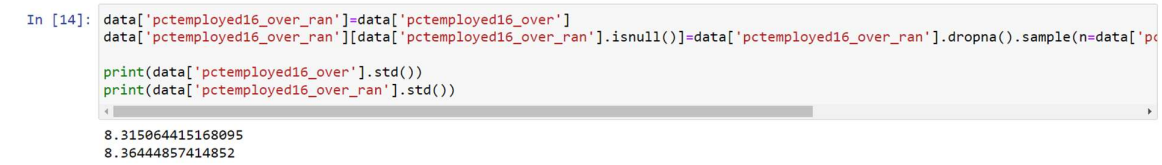
[pctemployed16_over = 69.140,

pctemployed16_over_mean = 65.69,

pctemployed16_over_median = 65.68

pctemployed16_over_mode = 65.69]

- So, we will go with random value imputation and fill the missing values in this feature, after filling the values check the standard deviation,



[pctemployed16_over = 8.31,

pctemployed16_over_ran = 8.32]

- There is no difference in the standard deviation value, so we will choose the "pctemployed16_over_ran" feature.

Pctprivatecoveragealone:

```
data['pctprivatecoveragealone_mean']=data['pctprivatecoveragealone'].fillna(data['pctprivatecoveragealone'].mean())
data['pctprivatecoveragealone_median']=data['pctprivatecoveragealone'].fillna(data['pctprivatecoveragealone'].median())
data['pctprivatecoveragealone_mode']=data['pctprivatecoveragealone'].fillna(48)

print(data['pctprivatecoveragealone_mode'].var())
print(data['pctprivatecoveragealone_mean'].var())
print(data['pctprivatecoveragealone_median'].var())
print(data['pctprivatecoveragealone'].var())
```

```
81.3732292594401
81.34028899639495
81.34998779463083
101.66701694009849
```

- After filling missing values, we should always check for variance, variance should not change much, but in the pctemployed16 over feature column is more variance, (Fig-2)

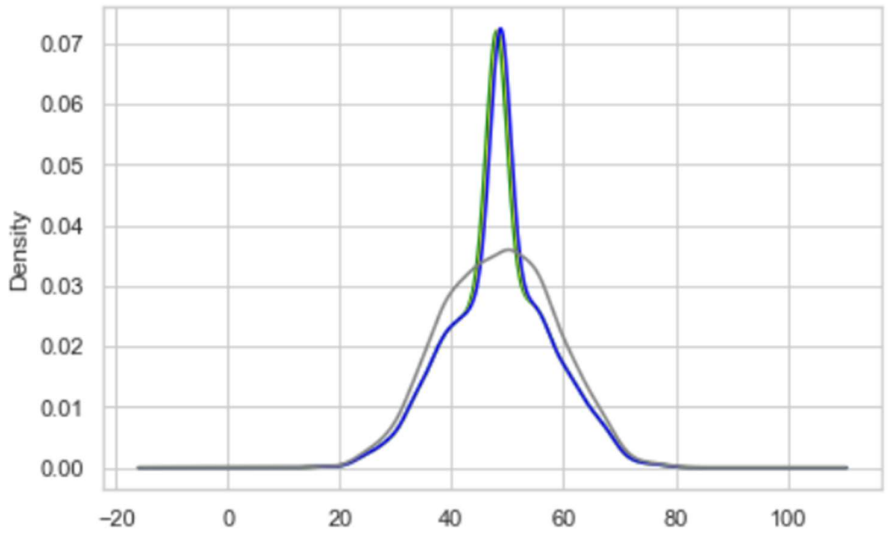


FIG-2

[pctprivatecoveragealone = 101,

pctprivatecoveragealone_mean = 81,

pctprivatecoveragealone_median = 81,

pctprivatecoveragealone_mode = 81]

- So, we will go with random value imputation and fill the missing values in this feature, after filling the values check the standard deviation,

```
data['pctprivatecoveragealone_ran']=data['pctprivatecoveragealone']
data['pctprivatecoveragealone_ran'][data['pctprivatecoveragealone_ran'].isnull()]=data['pctprivatecoveragealone_ran'].dropna().sa

print(data['pctprivatecoveragealone'].std())
print(data['pctprivatecoveragealone_ran'].std())
```

```
10.08300634434485
10.055861250809489
```

[pctprivatecoveragealone = 10,

pctprivatecoveragealone_ran = 10]

- There is no difference in the standard deviation value, so we will choose the "pctprivatecoveragealone_ran" feature.

C) State Wise Analysis of Death Rate:

- In our dataset, we have a feature on "geography" in this column there is states and county are there, so first we have split the state & county and make a new feature.

```
In [20]: data[['State','County','geography']].head(5)
```

Out[20]:

	State	County	geography
0	Washington	Kitsap County	Kitsap County, Washington
1	Washington	Kittitas County	Kittitas County, Washington
2	Washington	Klickitat County	Klickitat County, Washington
3	Washington	Lewis County	Lewis County, Washington
4	Washington	Lincoln County	Lincoln County, Washington

- In our dataset, we have 51 unique states and 1819 counties.

```
Number of states =  51
Number of Counties =  1819
```

- "Kentucky" has the highest number of deathrates in mean of 215, and followed by "Maine" has mean of 183. Some States have even more death rate than "Maine" which has less counties.

```
State
Kentucky          215.315833
Mississippi       202.837805
Tennessee         200.875789
Arkansas          200.090667
Louisiana         197.673438
West Virginia     196.710909
Oklahoma          193.949351
Alaska            193.416667
Alabama           192.728571
Missouri          189.694783
South Carolina    188.506522
Indiana           188.120652
Ohio              186.882759
Illinois          183.224510
Maine             183.150000
```

- In Kentucky, 'Powel County' has highest death rate of 292.5, ie) 35% more than the mean of target death rate.

	County	target_deathrate	birthrate
2597	Powell County	292.5	5.682217
2595	Perry County	280.8	4.955914
2175	Harlan County	277.1	3.257989
2563	Lee County	275.9	2.731512
2599	Robertson County	274.0	1.204819

- "Kansas" has highest birth rate 6.5, "Georgia" and "Illinois" have same birth rate like similarity in death rate.(Fig-3)

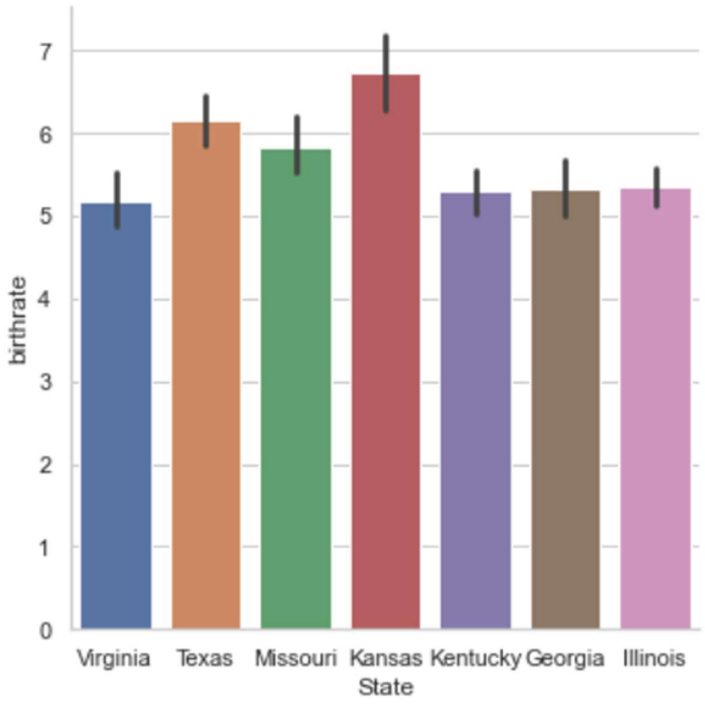


FIG-3

- "Texas" is having the higher incidence rate among the states, District of "Columbia" is having a less incidence rate, "Texas" and "Georgia" have counts greater than 150 whereas other features are below 150.(Fig – 4)

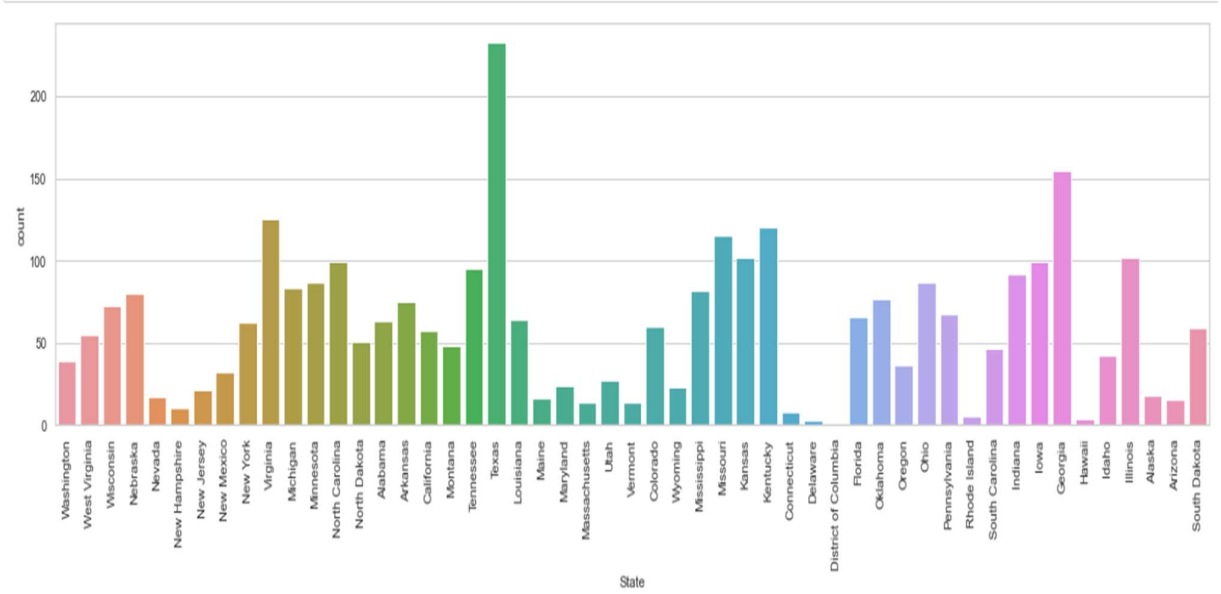


FIG - 4

D)Analyzing the Target variable:

- The target value will check before go for the model, and if it is normally distributed or not.

- Check the below target variable visualization is normally distributed.

(Fig -5)

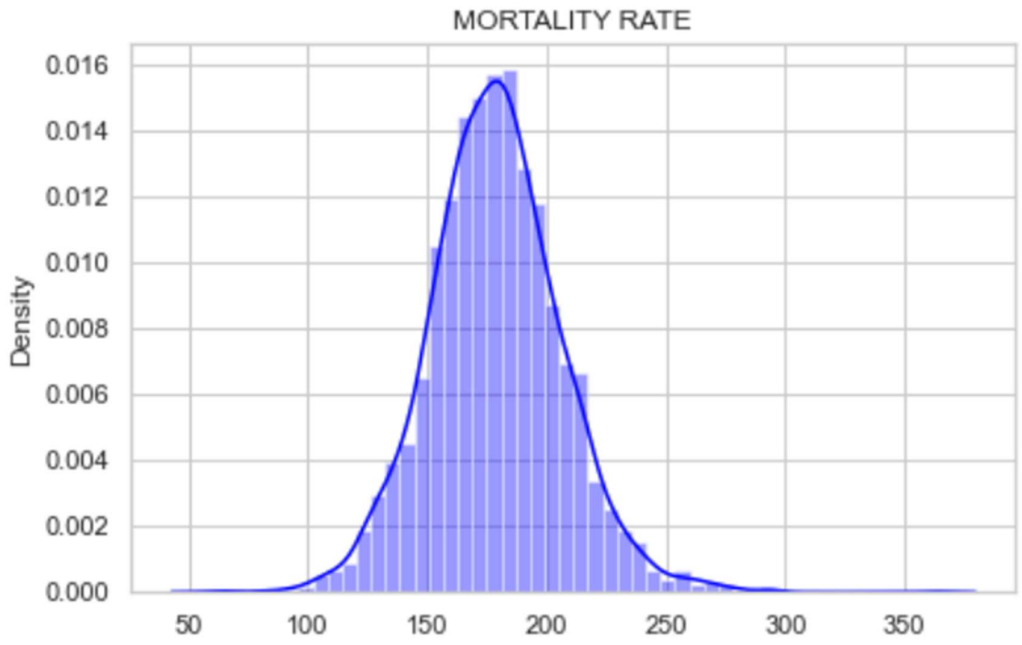


FIG-5

E)Checking Multicollinearity:

- Multicollinearity appears when there is strong correspondence among two or more independent variables in a multiple regression model,

- So, the multicollinearity is a problem because it produces Regression model results that are less reliable, so reduce the amount of multicollinearity features in our dataset,

```
{'avgdeathsperyear',
 'medianagefemale',
 'pctbachdeg25_over',
 'pctblack',
 'pctemployed16_over_ran',
 'pctempprivcoverage',
 'pctmarriedhouseholds',
 'pctprivatecoverage',
 'pctprivatecoveragealone_ran',
 'pctpubliccoverage',
 'pctpubliccoveragealone',
 'popest2015',
 'povertypercent'}
```

- The above features are not contributed for our target variable, because it is more correlated between in the independent variables,

- And these are the highly correlated features in our dataset, so we can drop the features and take the remaining features for model.

F)Correlation:

- To check the correlation for our independent and dependent variable, which features is more contribute for our dependent variables and select the more contribute features only.
- As per correlation the more correlate features is "incidencerate" = 0.44,
- The positive correlated features are below fig

incidencerate	0.449432
pctpubliccoveragealone	0.449358
povertypercent	0.429389
pcths25_over	0.404589
pctpubliccoverage	0.404572
pctunemployed16_over	0.378412
pcths18_24	0.261976
pctblack	0.257024
pctnohs18_24	0.088463
medianagefemale	0.012048
medianage	0.004375

- The Negative correlated features are below fig

medianagemale	-0.021929
studypercap	-0.022285
birthrate	-0.087407
avgdeathsperyear	-0.090715
popest2015	-0.120073
avganncount	-0.143532
pctwhite	-0.177400
pctasian	-0.186331
pctotherrace	-0.189894
percentmarried	-0.266820
pctempprivcoverage	-0.267399
pctbachdeg18_24	-0.287817
pctmarriedhouseholds	-0.293325
pctprivatecoveragealone_ran	-0.301203
pctemployed16_over_ran	-0.383243
pctprivatecoverage	-0.386066
medincome	-0.428615
pctbachdeg25_over	-0.485477

- For the comparison between the multicollinearity and correlation features, it is seen that the parameters 'pctpubliccoveragealone', 'povertypercent', 'pctblack', 'medianagefemale' are to be removed.
- The final features we extract the model are: incidencerate , pcths25_over, pctpubliccoverage, pctunemployed16_over, pcths18_24 ,pctnohs18_24 ,median age.

Univariate Analysis:

- After feature selection we have to check our features are normally distributes are not using histogram, and check the outliers using boxplot.
- Selected features are:
  ['incidencerate','medianage','pctnohs18_24',
  'pcths18_24','pcths25_over','pctunemployed16_over',
  'pctpubliccoverage','target_deathrate']

Fig 6 – incidencerate:

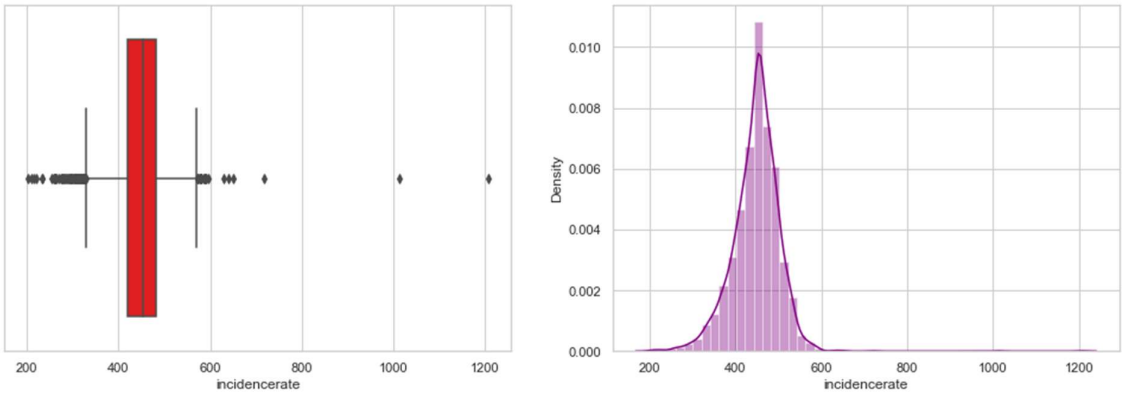


FIG -6

Fig 7- pcths25_over:

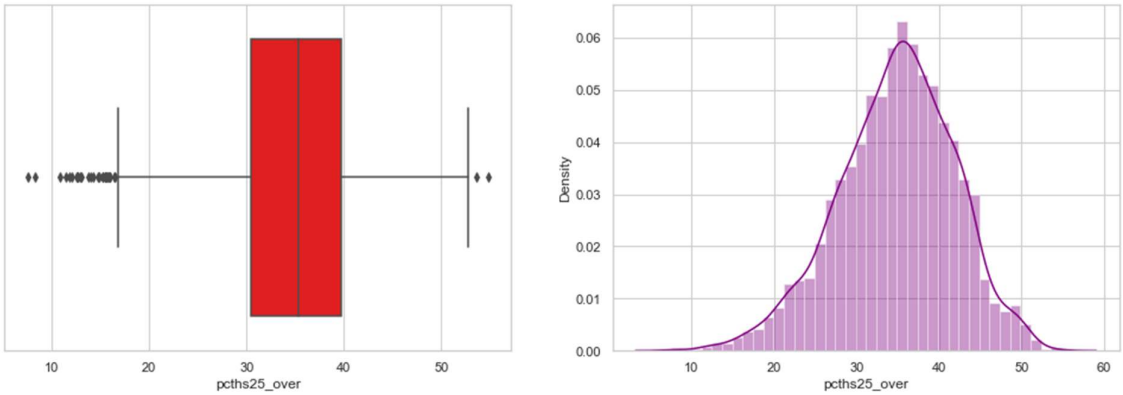


FIG -7

Fig 8 – pctpubliccoverage:

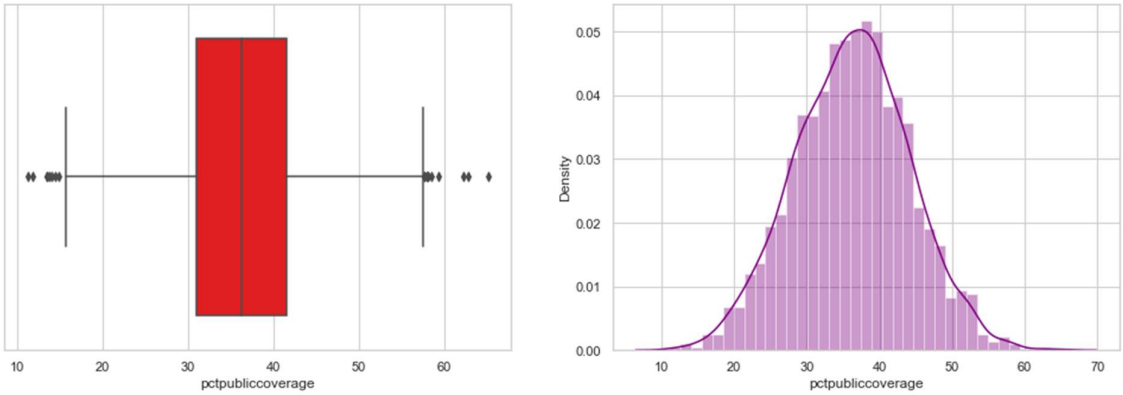


FIG-8

Fig 9 - pctunemployed16_over:

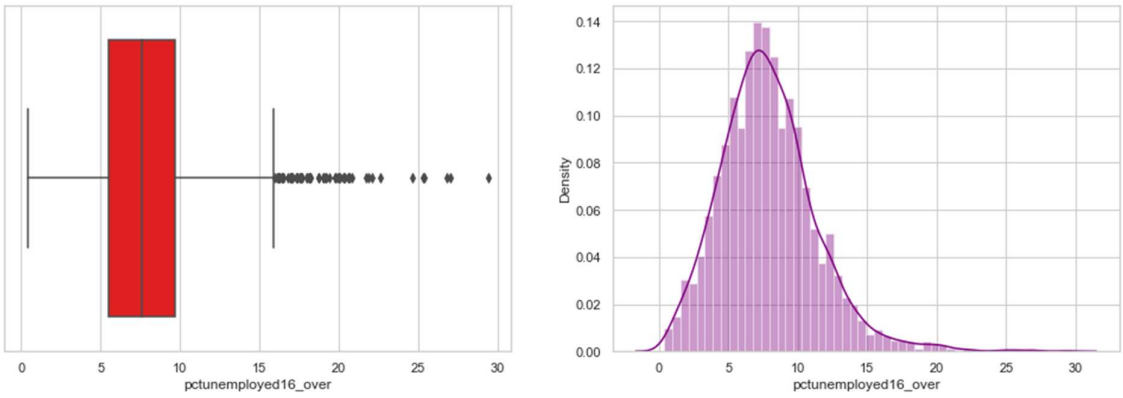


FIG – 9

Fig 10 - pcths18_24:

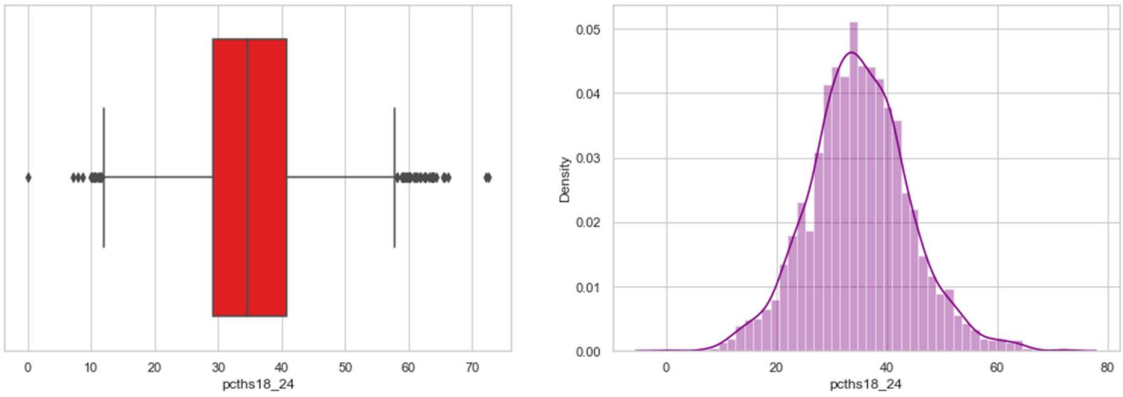


FIG -10

Fig 11- pctnohs18_24:

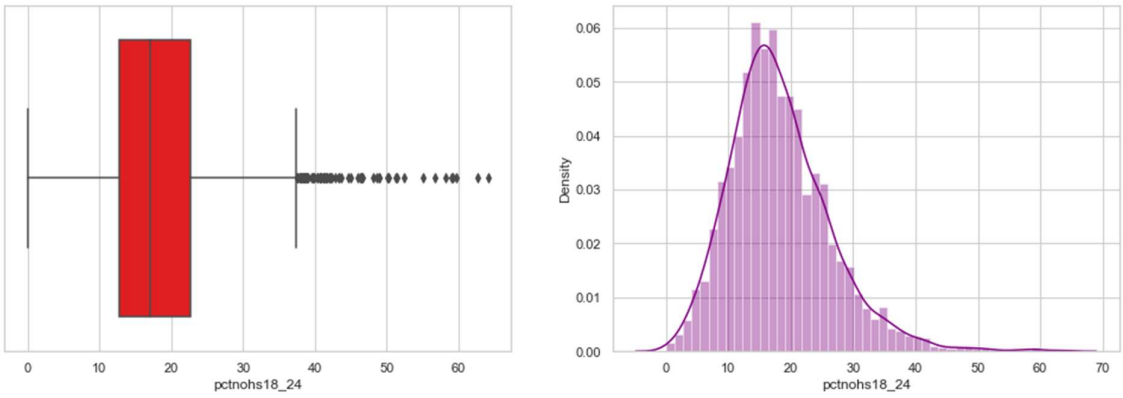


FIG -11

Fig 12 – medianage:

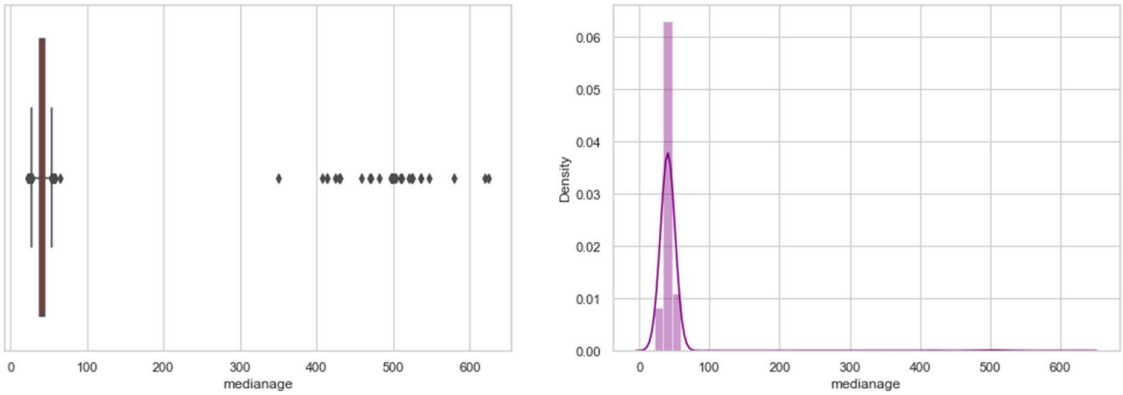


FIG -12

- From the above visualization, we infer that there are outliers in all features, and all the features are similar to normal distribution, so no issues first we replace the outliers.
- The interquartile method is used for outlier treatment.(FIG -13)

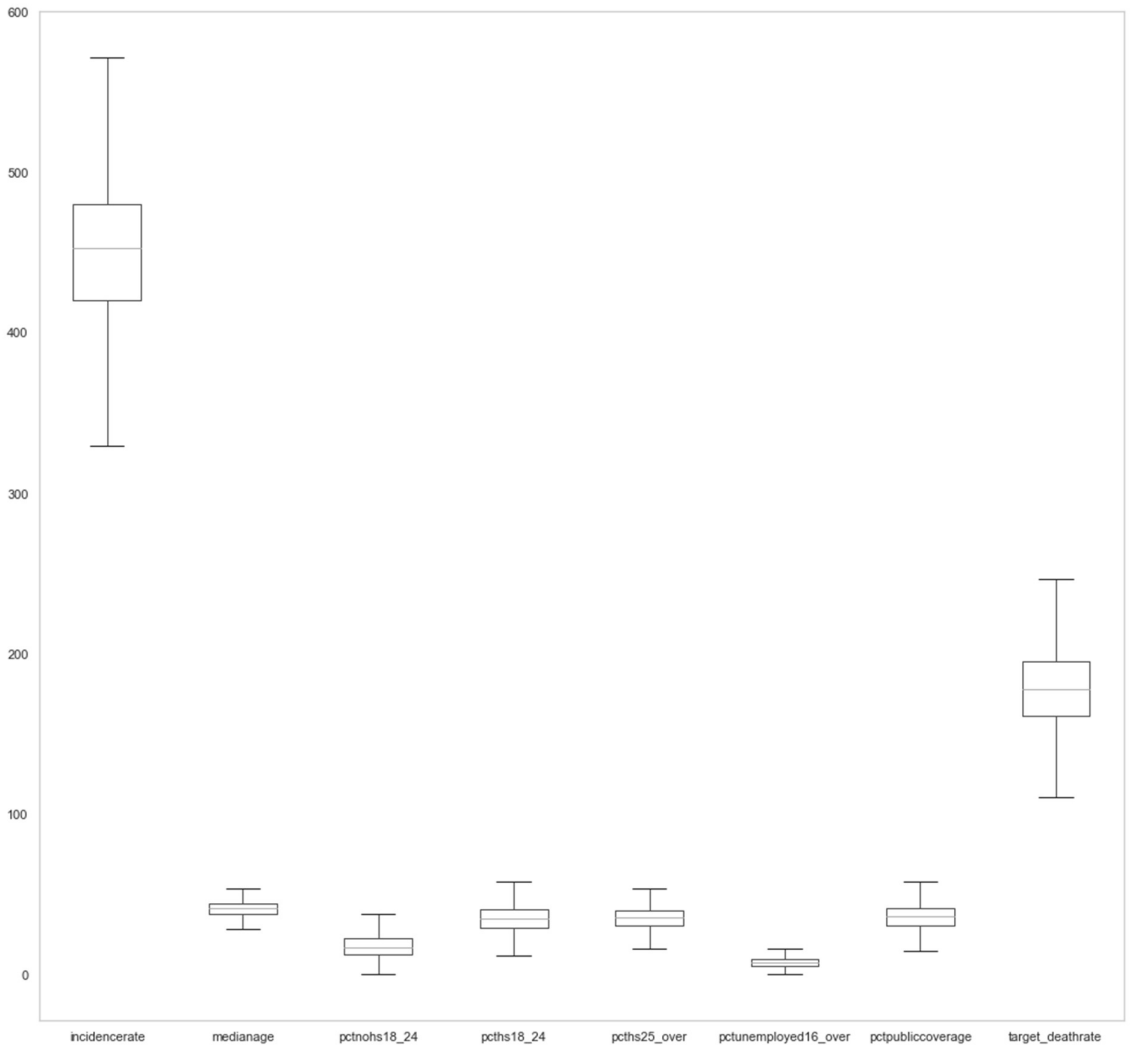


FIG -13

- After using interquartile method, the outliers are replaced using upper and lower limit of the data.

Bivariate Analysis :

- Bivariate Analysis is helpful to find the analysis of two variables to determine the relationships between them.
- So, we have to analysis our target variables and selected features and find the relationships between them.

G)Model 1 – Linear Regression:

- Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables, in our dataset the regression analysis is used to predict the **deathrate.**
- If a single independent variable is used to predict the value of numerical dependent variable is used for Simple Linear Regression.
- If more than one independent variable is used to predict the value of numerical dependent variable is used for Multiple Linear Regression.
- In our dataset, we have to predict the deathrate using Multiple Linear Regression.
- Select features and target variable is done our dataset, after to split the data into train and test using sklearn.

Scikit-Learn (SKlearn):

- Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.
- Using SKlearn, we have to split the data into x_train, x_test, y_train and y_test.
- The default split size in sklearn.model_selection.train_test_split is 75% training and 25% testing data split.

```
x2_train, x2_test, y2_train, y2_test = train_test_split(x2, y2, test_size=0.3, random_state=0)
print(x2_train.shape,x2_test.shape)
print(y2.shape, y2_train.shape, y2_test.shape)
```

```
(2132, 7) (915, 7)
(3047,) (2132,) (915,)
```

- The x_training shape is (2132,7) and x_test shape is (915,7) and same as y_training shape is (2132,) and y_test shape is (915,).
- After splitting the data, we have to fit the x_train and y_train data into linear regression model [**LinearRegression()**].
- The accuracy of the training data is 46%
  And accuracy of the testing data is 46%.

Evaluation Metrics for Regression Model:

1) Mean Absolute Error – MAE is a very simple metric which calculates the absolute difference between actual and predicted values.

Mean absolute Error is 14.637991914730396

2) Mean Square Error – MSE is most used and very simple metric with a little bit of change in mean absolute error. Mean Squared error states that finding the square difference between actual and predicted values.

Mean Square Error is 19.20530277169404

3) R squared (R2) – R2 score is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform.

R square value is 0.4632155298107906

- Scatter plot with our predicted target variable regression line. (FIG -14)

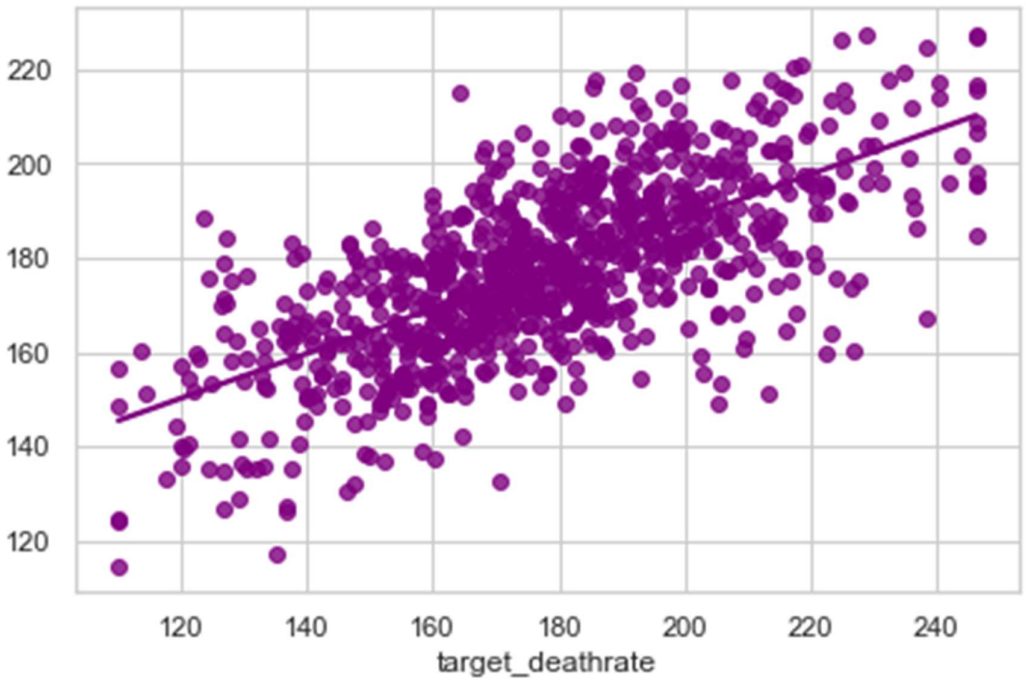


FIG -14

Evaluation Metrics	Accuracy
Mean absolute Error	14.637991914730396
Mean Square Error	19.20530277169404
R square	0.4632155298107906

- In this Linear Regression Model, the accuracy score is **46%** only, hence we try to tune our model better.

H) Model 2-Stochastic Gradient Descent:

- Gradient Descent is a generic optimization algorithm capable of finding optimal solutions to a wide range of problems.
- An important parameter of Gradient Descent is the size of the steps, determined by the learning rate hyperparameters.
- If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time, and if it is too high, we may jump the optimal value.
- Hence, in Stochastic Gradient Descent, a few samples are selected randomly instead of the whole dataset for each iteration.
- Using SGDRegressor() and fit the x_train and y_train,
- The accuracy of Stochastic Gradient Descent algorithm is also 46%, there is no improvement using the hyperparameter tuning.

Evaluation Metrics	Accuracy
Mean absolute Error	14.637991914730396
Mean Square Error	19.20530277169404
R square	0.4632155298107906

- The accuracy of SGD is 46%.

I) Source Code:

Import Required Packages:

```
# Data visualisation and Manipulation
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
from matplotlib.legend_handler import HandlerBase
import seaborn as sns
import scipy.stats as stats
from sklearn.preprocessing import PowerTransformer
from sklearn.decomposition import PCA

#configure
# sets matplotlib to inline and displays graphs below the corressponding cell.
%matplotlib inline
style.use('fivethirtyeight')
sns.set(style='whitegrid',color_codes=True)

#Regression
from sklearn.linear_model import LinearRegression

#Model selection
from sklearn.model_selection import train_test_split,cross_validate
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
```

```
#Evaluation metrics
from sklearn.metrics import mean_squared_log_error,mean_squared_error,
r2_score,mean_absolute_error


# Ignore the warnings
import warnings
warnings.filterwarnings('ignore')
```

Regression Model:

```
# Select Features
data=data[['incidencerate','medianage','pctnohs18_24',
        'pcths18_24','pcths25_over','pctunemployed16_over',
        'pctpubliccoverage','target_deathrate']]


# X and Y features
x2 = data.drop('target_deathrate',axis=1)
y2 = data['target_deathrate']


# Split the Dataset train and test
x2_train, x2_test, y2_train, y2_test = train_test_split(x2, y2, test_size=0.3,
random_state=0)
print(x2_train.shape,x2_test.shape)
print(y2.shape, y2_train.shape, y2_test.shape)


# Regression Fit.
reg_lin=LinearRegression()
reg_lin.fit(x2_train,y2_train)
pred=reg_lin.predict(x2_test)


print('Training data r-squared:', reg_lin.score(x2_train, y2_train))
print('Test data r-squared:', reg_lin.score(x2_test, y2_test))
```

```
#Evaluation Metrics
mse=np.sqrt(mean_squared_error(y2_test,pred))
print("The Mean Square Error is ",mse)

mae=mean_absolute_error(y2_test, pred)
print("The Mean absolute Error is ",mae)

r2=r2_score(y2_test, pred)
print("The R square value is",r2)

# Evaluating the model using adjusted R2 Evaluation Metric
# Defining the adjusted R² function
def adjusted_r2_score(actual, predictions, num_pred, num_samples):
    n = num_samples
    k = num_pred
    r2 = r2_score(actual, predictions)
    adjusted_r2 = 1 - ((1-r2) * ((n-1)/(n-k-1)))
    return adjusted_r2

# Initializing the model and fitting the model with train data
model = LinearRegression()
model.fit(x2_train,y2_train)

# Generating predictions over test data
predictions = model.predict(x2_test)

# Evaluating the model using Adjusted R² Evaluation Metric
num_samples = x2_test.shape[0]
num_predictors = x2_test.shape[1]
adjusted_r2_score(y2_test, predictions, num_predictors, num_samples)
```

Stochastic Gradient Descent:

```
# Import Package
from sklearn.linear_model import SGDRegressor
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

# SGD Regressor fit.
re_sdg=SGDRegressor()
re_sdg.fit(x2_train,y2_train)

#Evaluation Metrics
train_mae=mean_absolute_error(y2,re_sdg.predict(x2))
print("The Mean absolute error is:",train_mae)
sgd_pl=Pipeline([("feature_scaling",StandardScaler()),("sgd",SGDRegressor())])
sgd_pl.fit(x2,y2)
train_mae=mean_absolute_error(y2,sgd_pl.predict(x2))
print("The Mean absolute is:",train_mae)
score=sgd_pl.score(x2,y2)
print("R-squared",score)
```

Result & Conclusion:

- In this dataset, we have analysed and extracted the features to predict the target variable and find the deathrate of cancer.
- Using Regression model and further improved the hyperparameter tuning using Gradient descent, Stochastic Gradient and Mini batch gradient.
- The columns of pctsomecol18_24, pctemployed16_over, pctprivatecoveragealone has missing values with in them. Some of the columns having high percentage of missing values are dropped and remaining were filled with mean, median, mode and random values.
- In our dataset, we have 51 unique states and 1819 counties, "Kentucky" has the highest number of deathrates in mean of 215, and followed by "Maine" has mean of 183. Some States have even more death rate than "Maine" which has less counties.
- The poverty percent of the state "Kentucky" is 21.8(Which is large) and it has a smaller number of clinical trials.
- Median Income increases means death rate is also decreases, if suppose more income the death rate is also less.
- As poverty increases, the death rate is also increase simultaneously, Unemployed class may not get proper treatments, Literacy and employment people make aware of cancer and their treatment processes.
- From the selected features the Linear Regression Model was performed and it yields an accuracy is 46%.
- After using the hyperparameter tuning there is no improvement in our model, the accuracy remains same.