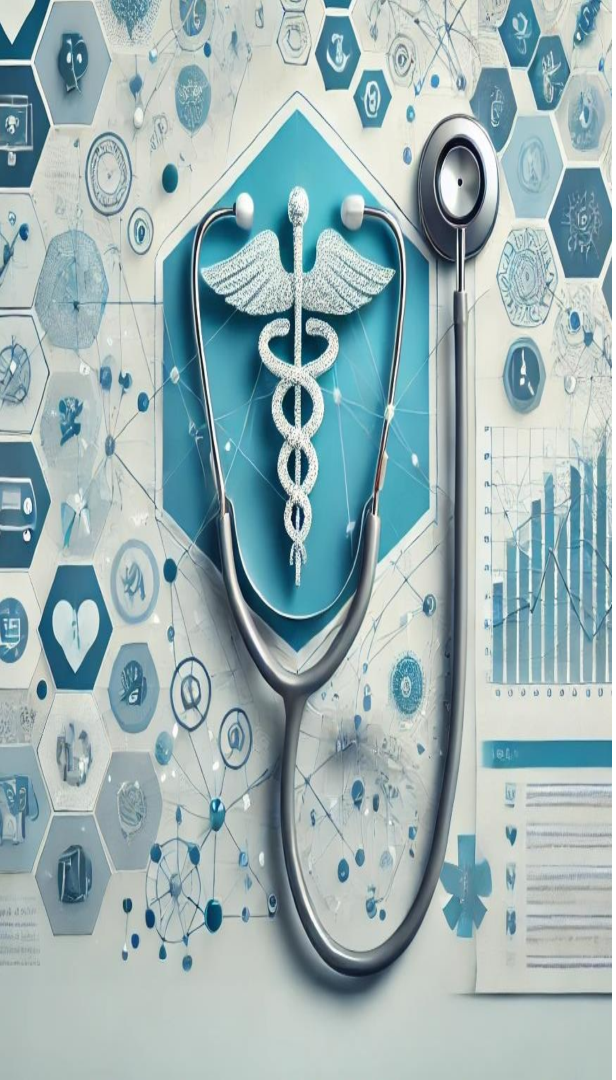
MEDICAL INSURANCE

GROUP 14 I BHAVANS VIVEKANANDA COLLEGE
ANSHUMAAN YADAV
SAMYUKTHA SATULURI
ABHINAV SHOURY
G. NAMRATH
UJJWAL KUMAR



ABSTRACT

The project focuses on building a predictive model to forecast insurance premiums based on customer profiles and risk factors, aiming to enhance accuracy in pricing policies and minimize loss ratios.

The study explores a range of machine learning algorithms, including Logistic Regression, KNN, Decision Trees, to predict premium amounts based on factors such as age, vehicle type, and claim history

OBJECTIVE

To identify the most suitable machine learning model for predicting insurance charges in the medical industry, with the goal of optimizing premium plans and improving pricing accuracy.

CONTENT

- Introduction
- Data Preprocessing
- Exploratory Data Analysis
- Data Modelling And Evaluation
- Summary

INTRODUCTION

The Medical Insurance Charges Dataset is an essential resource for analysing healthcare costs and identifying the factors that impact insurance premiums. Key attributes such as age, BMI, smoking habits, and region play a crucial role in determining the cost of medical coverage.
Technological advancements: The use of machine learning and data analytics has revolutionized the prediction of insurance premiums, enabling more precise forecasting and providing valuable insights for optimizing policy pricing and personalized healthcare plans.

DATA PREPROCESSING

DATA

DATASET – The dataset consists of 7 variables and 1339 records.

VARIABLES –

Categorical	Numerical
Region	Age
Sex	BMI
Smoker	Children
	Charges

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

DATA CLEANING

1. Data Overview:
- Inspected the dataset structure (`df.info()`, `df.describe()`, `df.shape`) and previewed the first few rows.

2. Missing Values:
- Checked for missing data using `df.isnull().sum()` (no missing values found).

3. Categorical and Numerical Analysis:
- Used `value_counts()` to check for outliers and anomalies in key columns like age, sex, smoker status, and charges.

DATA CLEANING

4. Encoding Categorical Variables:

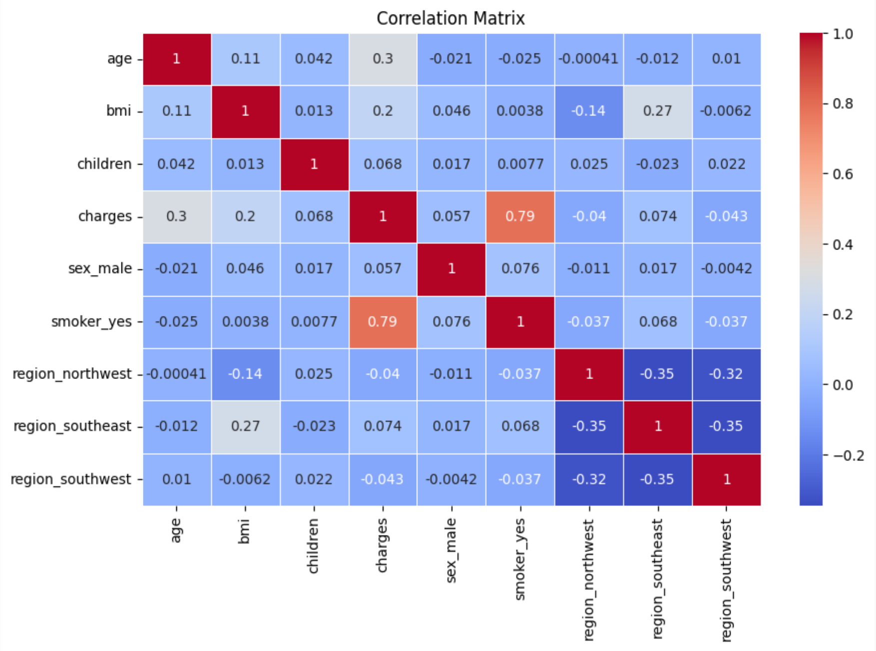
- Applied one-hot encoding (`pd.get_dummies`) to convert categorical variables into numeric form for modeling.

5. Multicollinearity Check:

- Calculated Variance Inflation Factor (VIF) to check for multicollinearity between numeric features.

EXPLORATORY DATA ANALYSIS

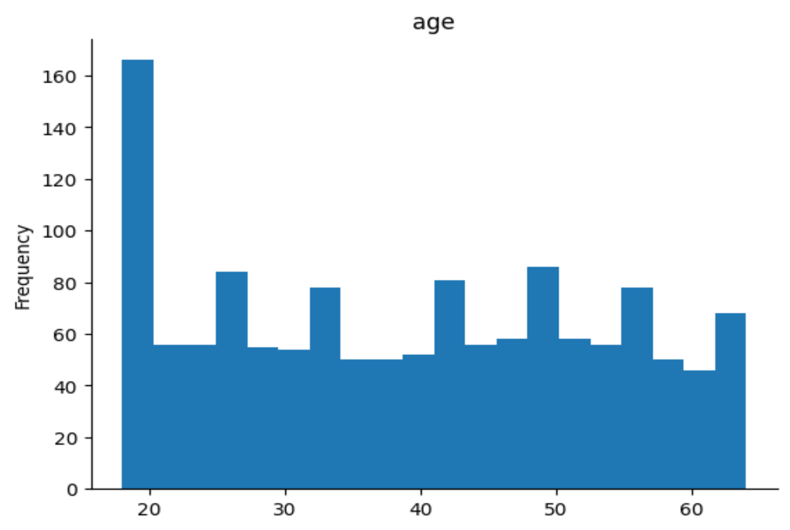
CORRELATION MATRIX

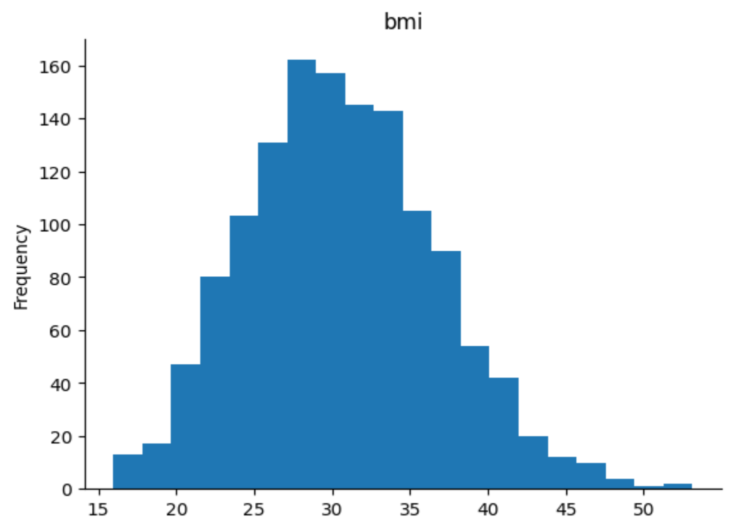


There is a strong correlation between –

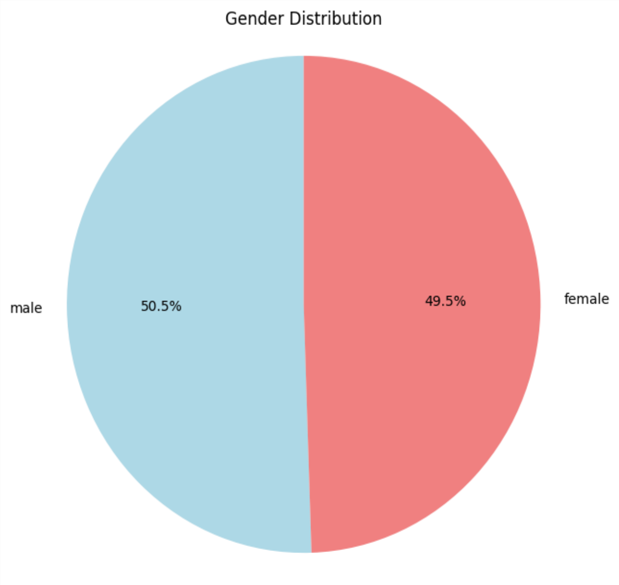
- Charges vs. Age: Older individuals tend to have higher insurance charges
- Charges vs. BMI: Individuals with higher BMI face increased insurance charges
- Charges vs. Smoking Status: Smokers are charged more for insurance.

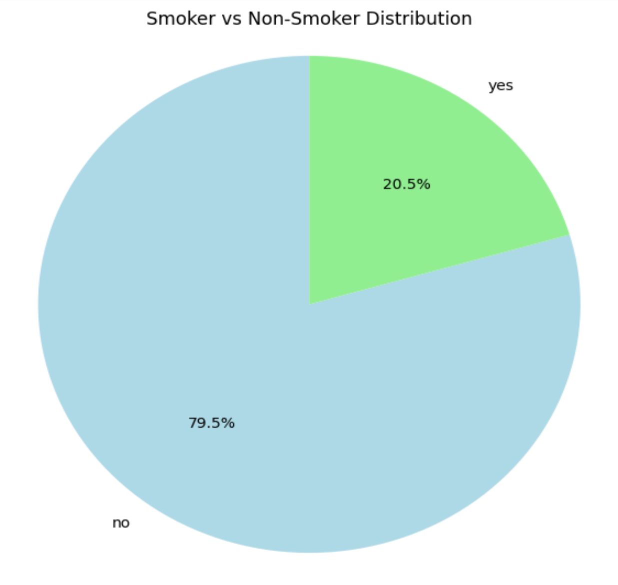
HISTOGRAM


age
Frequency
0
20
40
60
80
100
120
140
160
20
30
40
50
60

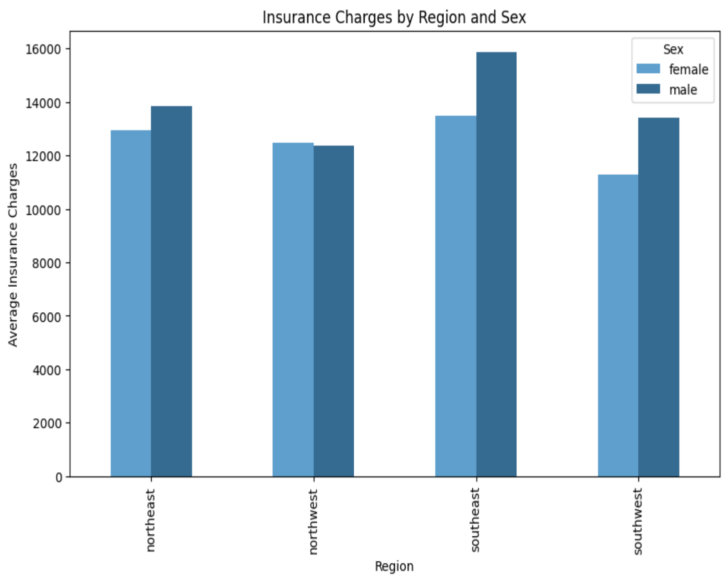

bmi
Frequency
0
20
40
60
80
100
120
140
160
15
20
25
30
35
40
45
50

PIE CHART

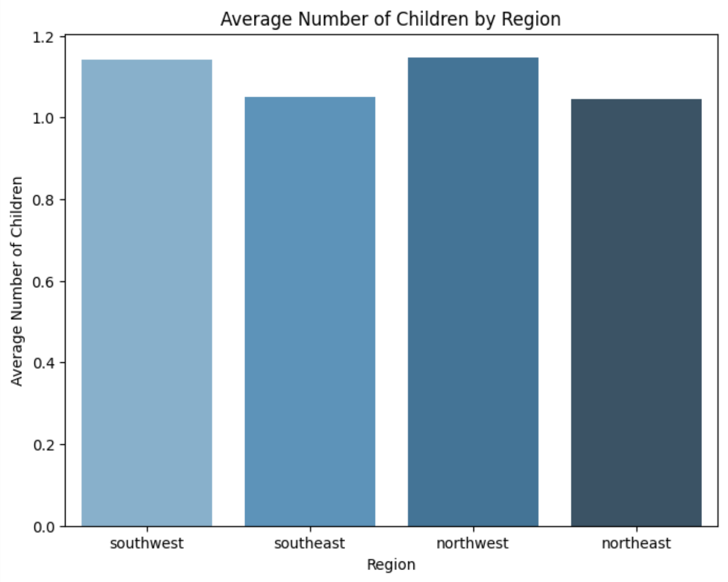

Gender Distribution
male
50.5%
49.5%
female


Smoker vs Non-Smoker Distribution
yes
20.5%
79.5%
no

BAR PLOT

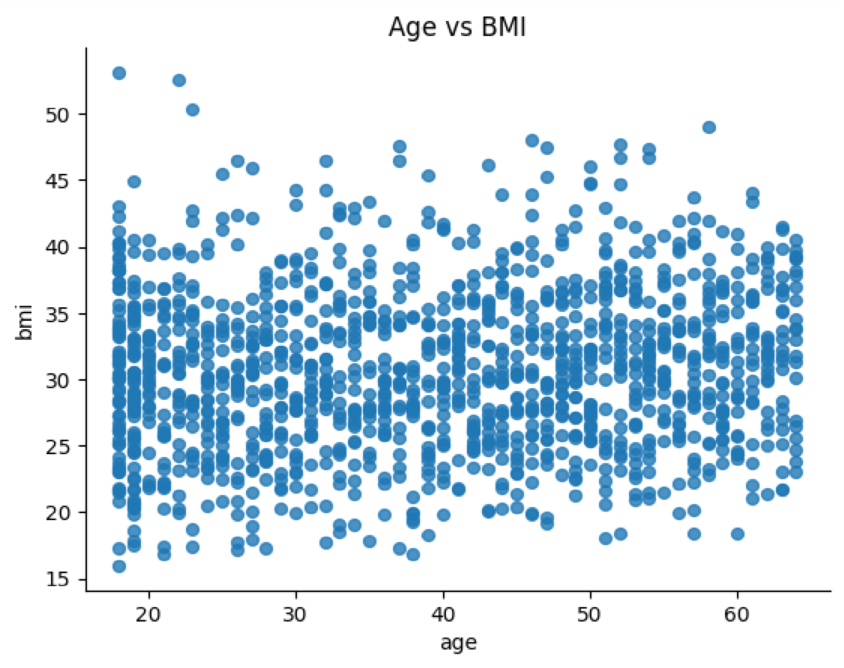


Insurance Charges by Region and Sex
Average Insurance Charges
16000
14000
12000
10000
8000
6000
4000
2000
0
Sex
female
male
northeast
northwest
southeast
southwest
Region



Average Number of Children by Region
Average Number of Children
1.2
1.0
0.8
0.6
0.4
0.2
0.0
southwest
southeast
northwest
northeast
Region

SCATTER PLOT



MACHINE LEARNING ALGORITHM

(CLASSIFICATION MODEL)

LOGISTIC REGRESSION

Train – test Split Ratio	Accuracy
0.20	91.04%
0.25	89.25%
0.35	89.9%

KNN CLASSIFICATION

Train – test Split Ratio	Accuracy
0.20	79.4%
0.25	77.31%
0.35	76.54%

DECISION TREE CLASSIFIER

Train – test Split Ratio	Accuracy
0.20	90.29%
0.25	90.74%
0.35	91.04%

ALGORITHM COMPARISON

Algorithm	Accuracy
Logistic Regression	91.04%
KNN Classification	77.31%
Decision Tree Classifier	91.04%

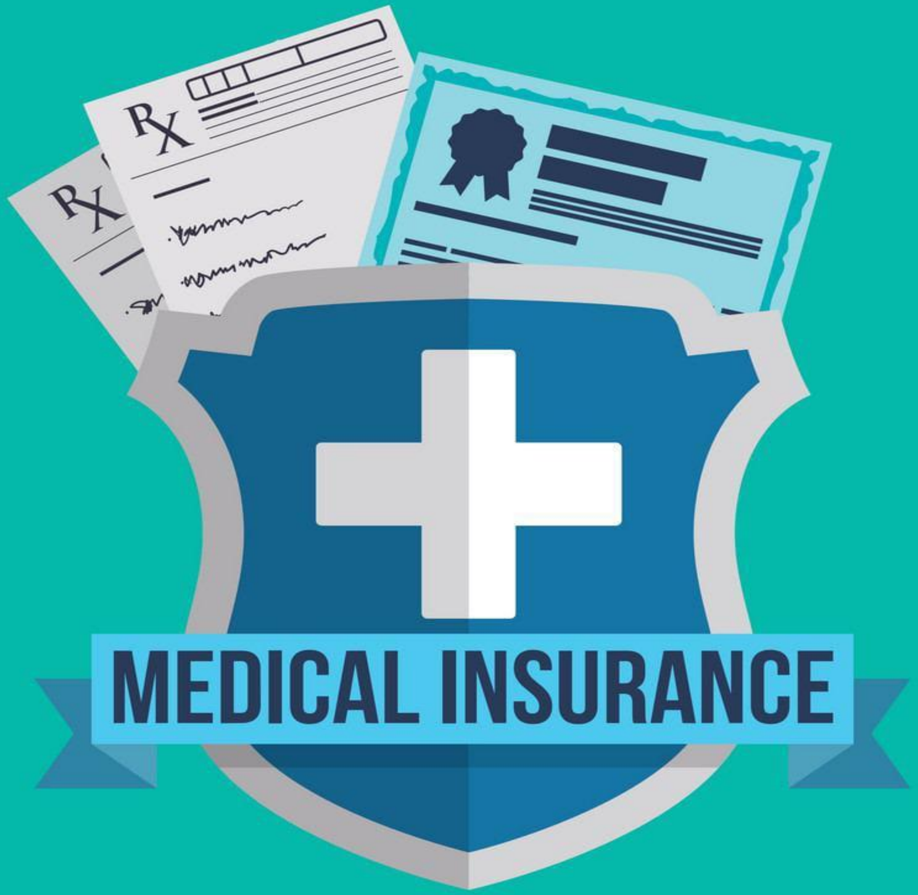
SUMMARY

- This study evaluates machine learning models for predicting medical insurance premiums, focusing on identifying the most accurate approach. The models compared include Logistic Regression, K-Nearest Neighbor (KNN), and Decision Tree Classifier.
- Logistic Regression and Decision Tree Classifier both achieved the highest accuracy at 91.04%, making them equally effective in this context.
- KNN Classification had the lowest performance, with an accuracy of 77.31%.
- The results suggest that Logistic Regression and Decision Tree models are equally suitable for predicting insurance premiums, while KNN underperforms in comparison

Thank You!

- ANSHUMAAN YADAV
- SAMYUKTHA SATULURI
- ABHINAV SHOURY
- G NAMRATH
- UJJWAL KUMAR

APPENDIX



Loading the Dataset

```
df=pd.read_csv('/content/Insurance Premium.csv')
df
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

1338 rows × 7 columns

Null Values

Checking for the data type

```
df.isnull().sum()
```

	0
age	0
sex	0
bmi	0
children	0
smoker	0
region	0
charges	0

dtype: int64

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
None
```

Calculating Variance Inflation Factor (VIF) for Feature Selection in Regression Analysis

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Select only numeric columns (ignoring any non-numeric)
df_numeric = df_dummy.select_dtypes(include=[np.number])

# Create DataFrame to store VIF values
vif_data = pd.DataFrame()
vif_data["Feature"] = df_numeric.columns
vif_data["VIF"] = [variance_inflation_factor(df_numeric.values, i) for i in range(df_numeric.shape[1])]

print(vif_data)
```

```
    Feature       VIF
0       age  8.098132
1       bmi  8.044400
2  children  1.800015
3   charges  2.473524
```

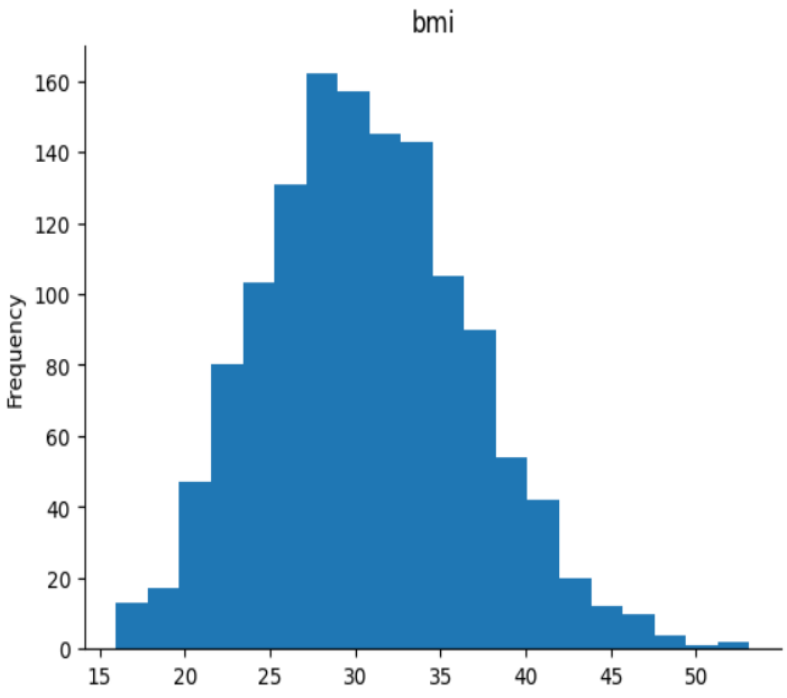
Splitting Dataset into Training and Testing Sets for Regression Analysis

```
X = df_dummy.drop(['charges'],axis=1)
y = df_dummy['charges']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.4, random_state=42)
print(X_train.shape,y_train.shape,X_test.shape,y_test.shape)
```

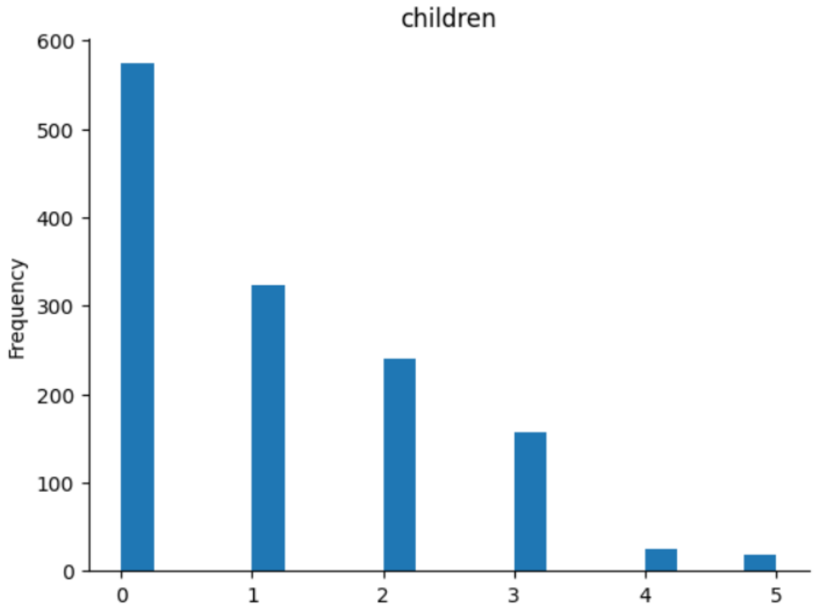
```
(802, 8) (802,) (536, 8) (536,)
```

Plots- HIST

```
df['bmi'].plot(kind='hist', bins=20, title='bmi')
plt.gca().spines[['top', 'right',]].set_visible(False)
```




```
from matplotlib import pyplot as plt
df['children'].plot(kind='hist', bins=20, title='children')
plt.gca().spines[['top', 'right',]].set_visible(False)
```



Logistic Regression

```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

df1 = df_dummy.copy()

charges_med = df1['charges'].median()
df1['catcharges'] = [1 if x > charges_med else 0 for x in df1['charges']]

df1['catcharges'].value_counts(normalize = True)
df1 = df1.drop('charges', axis=1)

x = df1.drop(['catcharges'],axis=1)
Y = df1['catcharges']
x_train, x_test, Y_train, Y_test = train_test_split(x, Y, test_size=.2, random_state=42)
print(x_train.shape,Y_train.shape,x_test.shape,Y_test.shape)
```

```
(1070, 8) (1070,) (268, 8) (268,)
```

```python
model3=LogisticRegression()
model3.fit(x_train,Y_train)
```

```python
model3.coef_
```

```
array([[ 0.14663096,  0.03259212,  0.05972964, -0.23076158,  6.1251742 ,
        -0.32717959, -0.55202742, -0.65595449]])
```

```python
Y_pred = model3.predict(x_test)
Y_pred
```

```
array([1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0,
       1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1,
       0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1,
       0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1])
```

```python
# binary, multiclass

from sklearn.metrics import classification_report
classification_rep = classification_report(Y_test, Y_pred)
print(classification_rep)
```

```
              precision    recall  f1-score   support

           0       0.94      0.90      0.92       146
           1       0.88      0.93      0.90       122

    accuracy                           0.91       268
   macro avg       0.91      0.91      0.91       268
weighted avg       0.91      0.91      0.91       268
```

K-Nearest Neighbor

```
[ ] df_dummy.shape
(1338, 9)
```

```
[ ] print(1338 ** 0.5)
36.578682316343766
```

```
[ ] from sklearn.neighbors import KNeighborsClassifier
    model2=KNeighborsClassifier(n_neighbors=37)
```

```
model2.fit(x_train, Y_train)
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=37)
```

```
Y_pred = model2.predict(x_test)
Y_pred
```

```
array([0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1,
       1, 0, 1, 1])
```

```
[ ] accuracy_score(Y_test,Y_pred)
0.7910447761194029
```

```
[ ] cm2 = confusion_matrix(Y_test,Y_pred)
    cm2
array([[135,  11],
       [ 45,  77]])
```

```
[ ] metrics.mean_absolute_error(Y_test,Y_pred)
0.208955223880597
```

Decision Tree

```
[ ] feature_cols = ['age', 'bmi', 'children', 'sex_male','smoker_yes','region_northwest','region_southeast','region_southwest']
    s = df1[feature_cols] # Features
    r = df1['catcharges']
    s_train, s_test, r_train, r_test = train_test_split(s, r, test_size=0.2, random_state=1) # 70% training and 30% test
```

```
[ ] from sklearn.tree import DecisionTreeClassifier

    clf = DecisionTreeClassifier()

    clf = clf.fit(s_train,r_train)
```

```
r_pred = clf.predict(s_test)
print("Accuracy:",metrics.accuracy_score(r_test, r_pred))
```

```
Accuracy: 0.9029850746268657
```

```
[ ] clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

    clf = clf.fit(s_train,r_train)

    r_pred = clf.predict(s_test)

    print("Accuracy:",metrics.accuracy_score(r_test, r_pred))
```

```
Accuracy: 0.917910447761194
```

```
[ ] clf = DecisionTreeClassifier(criterion="gini", max_depth=2)

    clf = clf.fit(s_train,r_train)

    y_pred = clf.predict(s_test)

    print("Accuracy:",metrics.accuracy_score(r_test, r_pred))
```

```
Accuracy: 0.917910447761194
```

```
[ ] clf = DecisionTreeClassifier(criterion="gini", max_depth=2)

    clf = clf.fit(s_train,r_train)

    y_pred = clf.predict(s_test)

    print("Accuracy:",metrics.accuracy_score(r_test, r_pred))
```

```
Accuracy: 0.917910447761194
```

```
clf = DecisionTreeClassifier(criterion="gini", max_depth=3)

clf = clf.fit(s_train,r_train)

y_pred = clf.predict(s_test)

print("Accuracy:",metrics.accuracy_score(r_test, r_pred))
```

```
Accuracy: 0.917910447761194
```

Random Forest

```
[ ] from sklearn.datasets import load_iris
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import classification_report, confusion_matrix
    from sklearn.tree import plot_tree

    from matplotlib import pyplot as plt
```

```
[ ] A = df1[['age', 'bmi', 'children', 'sex_male', 'smoker_yes', 'region_northwest', 'region_southeast', 'region_southwest']]
    B = df1['catcharges']
```

```
[ ] A_train, A_test, B_train, B_test = train_test_split(A, B, test_size=0.2)
```

```
print(A_train.shape)
print(B_train.shape)
print(A_test.shape)
print(B_test.shape)
```

```
(1070, 8)
(1070,)
(268, 8)
(268,)
```

```
[ ] rf = RandomForestClassifier()
```

```
[ ] rf.fit(A_train, B_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
[ ] B_pred = rf.predict(A_test)
```

```
print(classification_report(B_test, B_pred))
print(confusion_matrix(B_test, B_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93       135
           1       0.94      0.90      0.92       133

    accuracy                           0.93       268
   macro avg       0.93      0.93      0.93       268
weighted avg       0.93      0.93      0.93       268

[[128   7]
 [ 13 120]]
```