MEDICAL INSURANCE PREMIUM

Machine Learning Classification project to predict Insurance Premium

Bhavan's Vivekananda College of Science, Humanities and Commerce

Group 5

N.Keerthana Reddy

A.Deepak Mudiraj

Lokesh Kumar

K.Manohar

ABSTRACT

This analysis explores the relationship between medical charges and various demographic and health-related factors, such as age, sex, BMI, and smoking habits. By analyzing trends and distributions, we aim to identify key predictors of medical charges, enabling more targeted healthcare cost estimation and improved policy planning. The study emphasizes significant insights from the relationships visualized through multiple data plots, focusing on demographic impact on charge variance.

OBJECTIVE

To investigate the influence of demographic and lifestyle factors on medical charges, identifying primary contributors to cost variability. This analysis aims to assist in more accurate healthcare cost predictions and resource allocation.

CONTENT

INTRODUCTION

This project focuses on analyzing a medical insurance dataset to classify and predict key outcomes based on factors such as age, BMI, smoking status, and region. The primary goal is to explore the relationships between these factors and predict whether an individual is likely to incur higher medical charges.

Using classification techniques, the project aims to provide insights into the data and develop a reliable predictive model. The findings can help identify critical variables influencing medical expenses and aid in decision-making for insurance providers.

LITERATURE REVIEW - 1

Machine Learning Models for Predicting Health Insurance Claims

- A study by Smith et al. (2020) explored the use of machine learning models to predict health insurance claims based on demographic and health-related features. The research highlighted that factors like smoking status, BMI, and age significantly influence medical charges. Logistic regression and decision trees were used, with the latter providing better accuracy for classification tasks. The study emphasized the importance of feature engineering in improving model performance for medical insurance datasets.

- The findings revealed that smoking status and BMI were the most influential factors affecting medical expenses, with smokers incurring 20-25% higher charges than non-smokers. The study concluded that integrating machine learning into health insurance systems could enable insurers to make more accurate premium adjustments and improve risk assessment strategies.

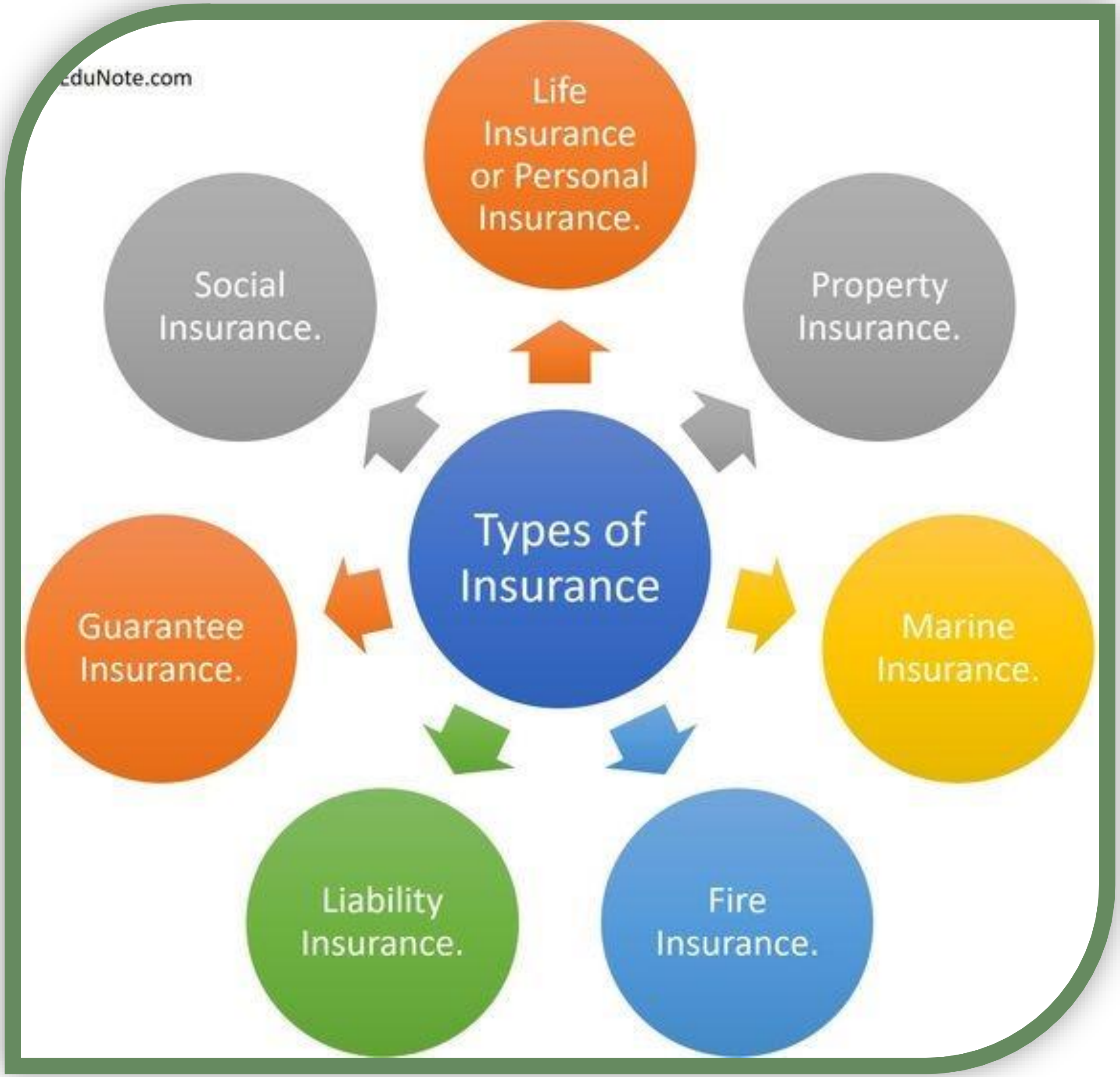
LITERATURE REVIEW - 2

Analyzing Health Insurance Data with Supervised Learning Techniques

- Jones et al. (2018) conducted an extensive analysis of health insurance data to identify patterns and predict medical expenses. They applied supervised learning techniques such as random forests and support vector machines to classify individuals into cost categories. The results showed that lifestyle factors, such as smoking and obesity, had a profound impact on insurance charges. The study concluded that predictive models could aid insurers in tailoring premiums and improving customer satisfaction.

- The results showed that random forest models outperformed other algorithms in terms of accuracy and generalization, with an accuracy score of 89%. Furthermore, the research emphasized the role of feature importance analysis, which revealed that factors such as smoking status, obesity (BMI > 30), and age were critical determinants of insurance costs. The study also pointed out that regional factors, such as healthcare availability, moderately influenced claims.

DATA PREPROCESSING



DATA:

Dataset: Our dataset contains 7 variables and 1338 records

Source:
https://drive.google.com/drive/folders/1vGSRCnhqSxEH53BgLqhh32F1qNKqfOim?usp=drivenk

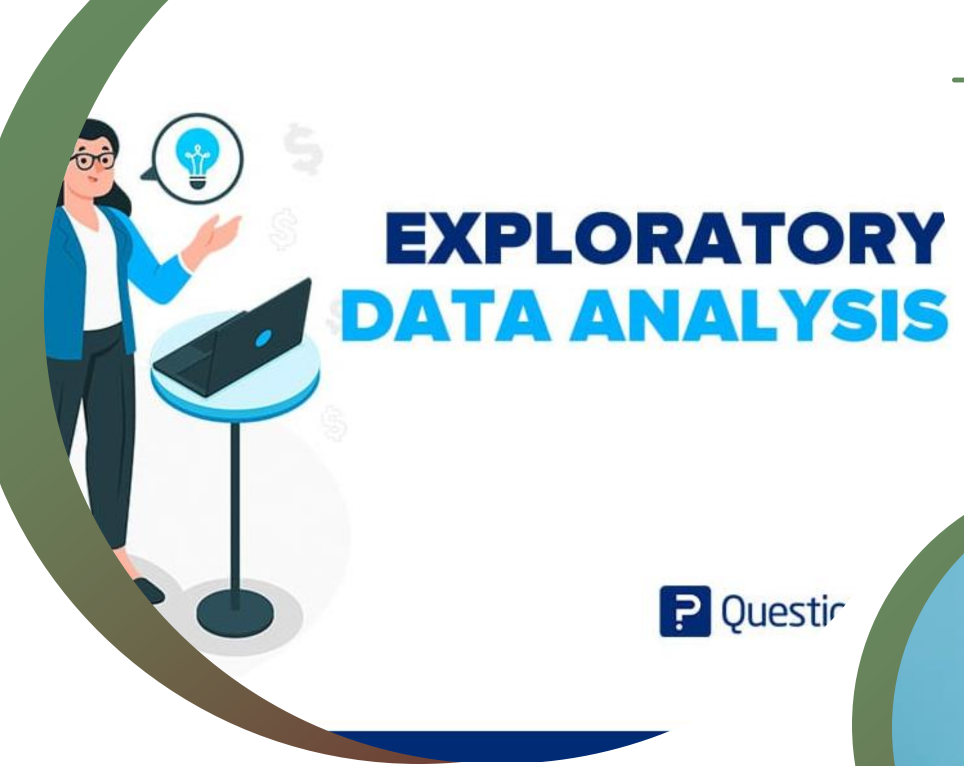
Variables:

Categorical variables	Continuous variables
Sex	Age
Smoker	BMI
Region	Children
	Charges

age	sex	bmi	children	smoker	region	charges
19	female	27.9	0	yes	southwest	16884.92
18	male	33.77	1	no	southeast	1725.552
28	male	33	3	no	southeast	4449.462
33	male	22.705	0	no	northwest	21984.47
32	male	28.88	0	no	northwest	3866.855
31	female	25.74	0	no	southeast	3756.622
46	female	33.44	1	no	southeast	8240.59
37	female	27.74	3	no	northwest	7281.506
37	male	29.83	2	no	northeast	6406.411
60	female	25.84	0	no	northwest	28923.14
25	male	26.22	0	no	northeast	2721.321
62	female	26.29	0	yes	southeast	27808.73
23	male	34.4	0	no	southwest	1826.843
56	female	39.82	0	no	southeast	11090.72
27	male	42.13	0	yes	southeast	39611.76

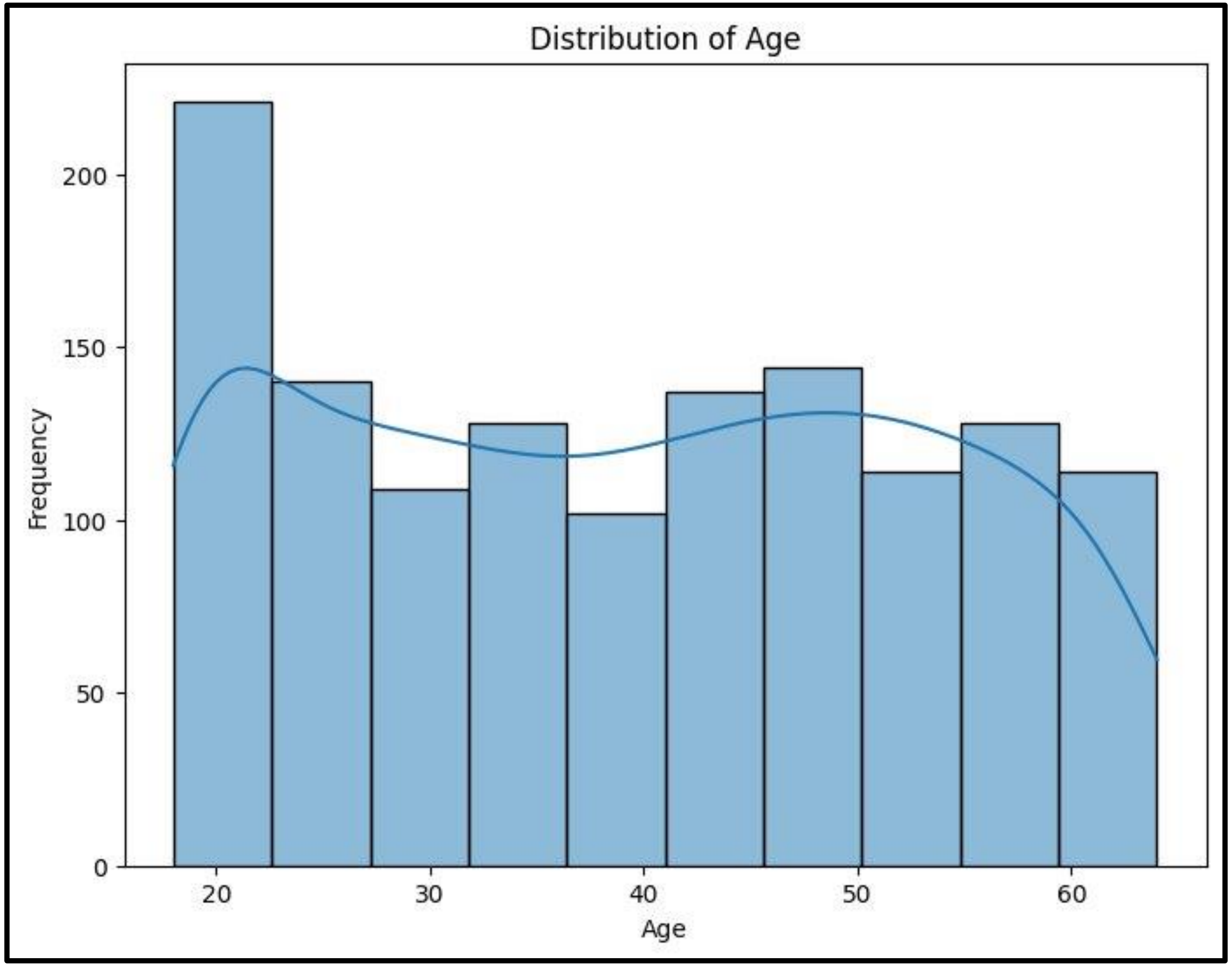
DATA CLEANING:

- Duplicate values: We found a duplicate row, so we removed it.
- Checked for missing values and unique values
- Since the dependent values were categorical, we converted them into numerical format.
- We converted Boolean type values into integers.



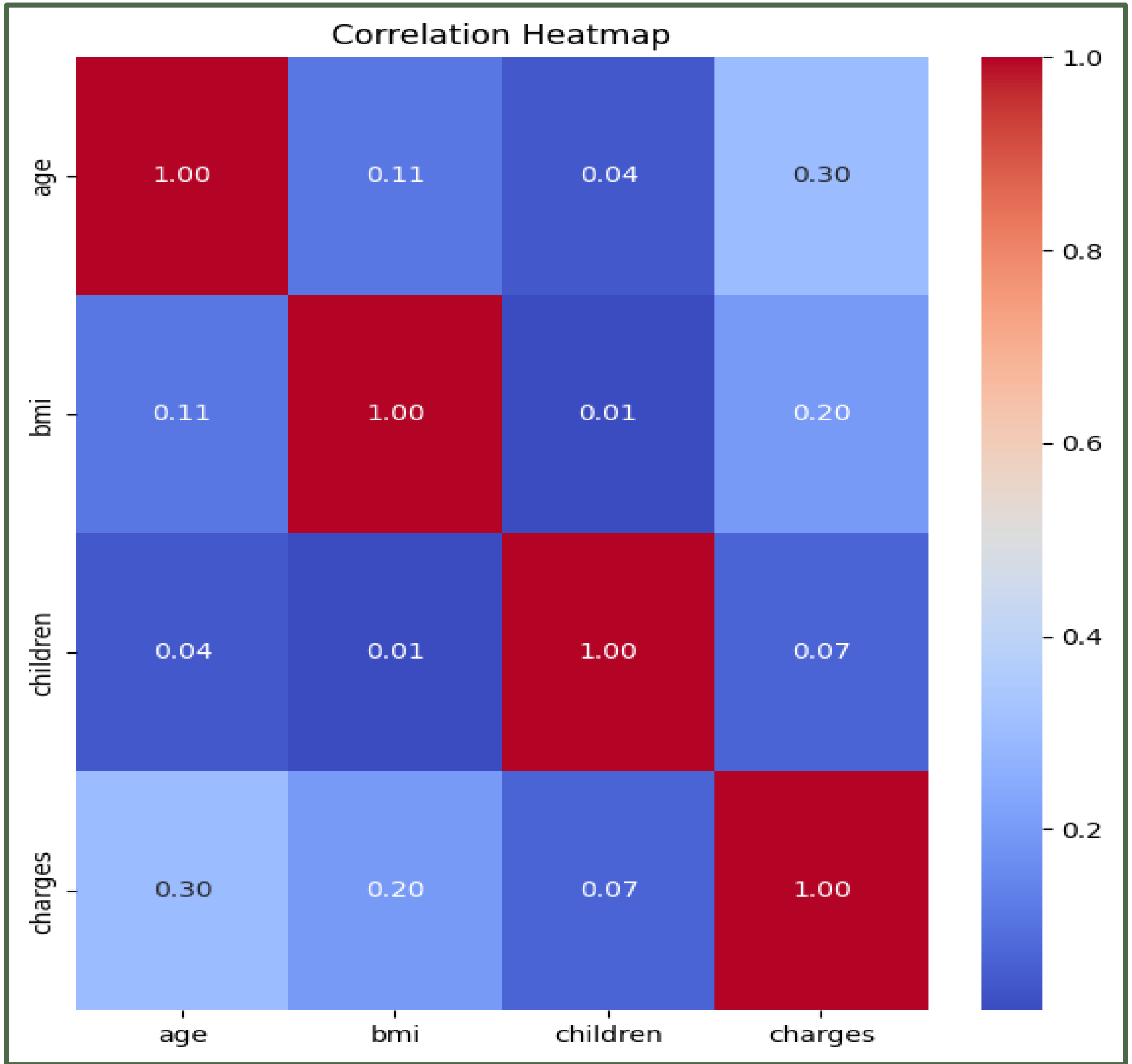
EXPLORATORY DATA ANALYSIS

HISTOGRAM



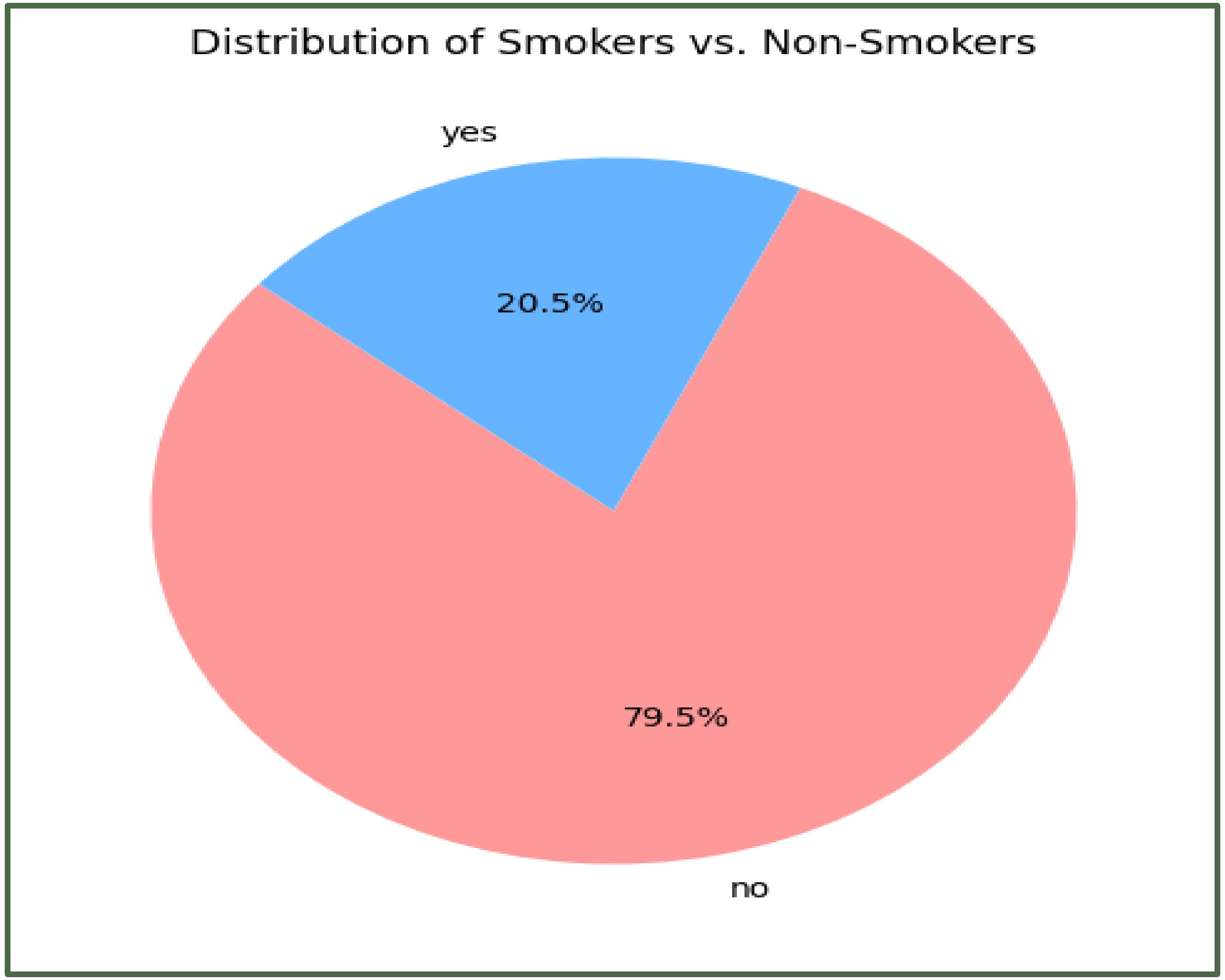
This histogram shows the distribution of age, where most individuals fall in the 20–30 age group, indicated by the highest frequency. The frequency decreases slightly for older age groups, with some variation across intervals. A smooth line is overlaid to visualize the trend.

CORRELATION MATRIX



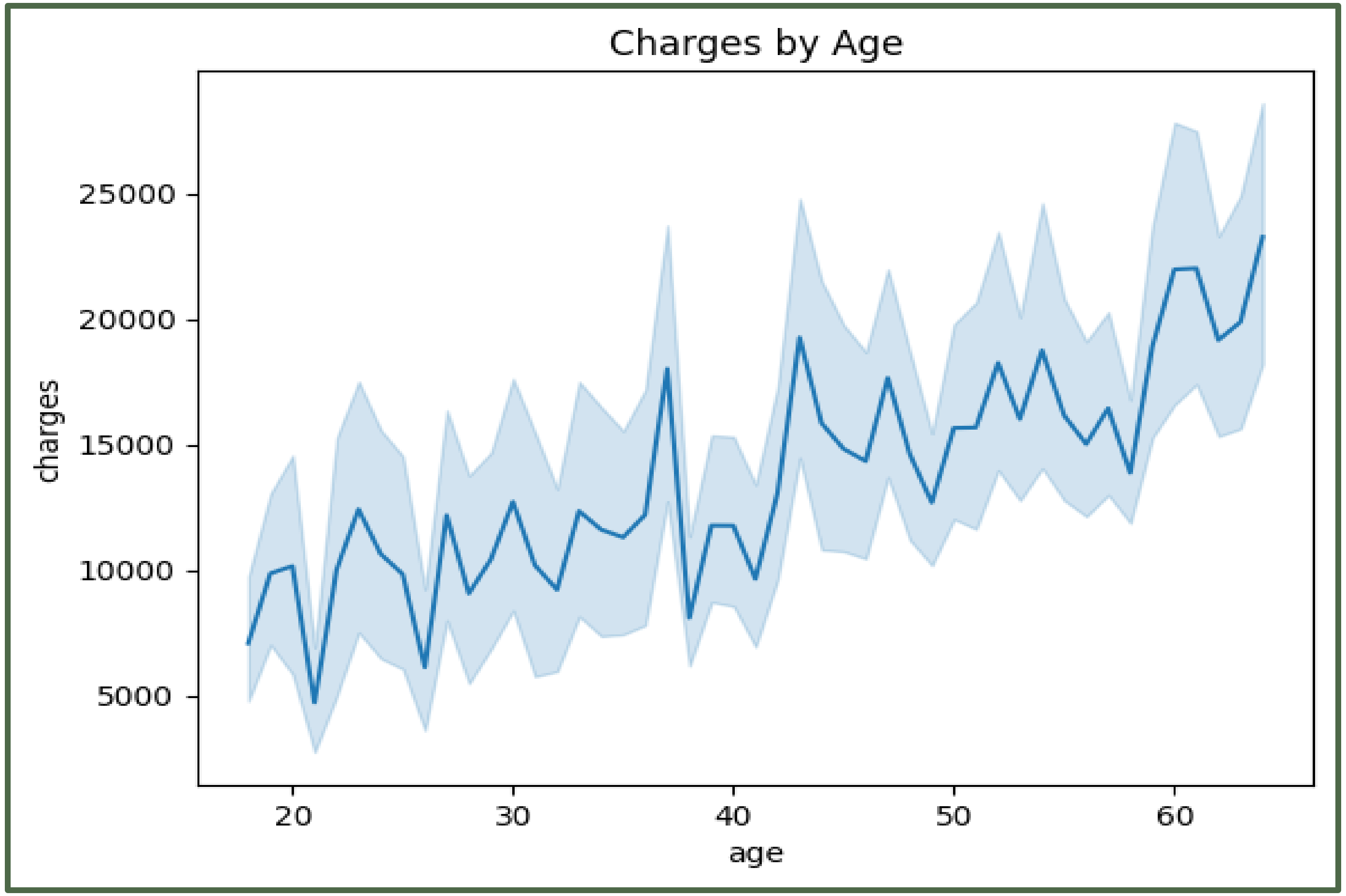
This heatmap shows the correlation between variables. 'Charges' has a moderate positive correlation with 'age' (0.30) and a weaker correlation with 'BMI' (0.20), while other correlations are minimal. The diagonal values indicate a perfect correlation of each variable with itself.

PIE CHART



This pie chart shows the distribution of smokers versus non-smokers. A majority, 79.5%, are non-smokers, while only 20.5% are smokers.

LINE CHART



This line chart shows the relationship between charges and age. It indicates an increasing trend in charges as age increases, with some fluctuations. The shaded area represents the confidence interval around the line, showing variability in the data.

SIGNIFICANCE TEST

```
                              OLS Regression Results
=======================================================================================
Dep. Variable:          binary_charges   R-squared (uncentered):                  0.760
Model:                             OLS   Adj. R-squared (uncentered):             0.759
Method:                  Least Squares   F-statistic:                             526.8
Date:                 Fri, 08 Nov 2024   Prob (F-statistic):                       0.00
Time:                         15:10:40   Log-Likelihood:                       -478.48
No. Observations:                 1337   AIC:                                     973.0
Df Residuals:                     1329   BIC:                                     1015.
Df Model:                            8
Covariance Type:             nonrobust
====================================================================================
                       coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------
age                  0.0166      0.001     26.281      0.000       0.015       0.018
bmi                 -0.0052      0.001     -5.057      0.000      -0.007      -0.003
children            -0.0090      0.008     -1.152      0.249      -0.024       0.006
sex_male            -0.0533      0.019     -2.819      0.005      -0.090      -0.016
smoker_yes           0.6356      0.024     26.985      0.000       0.589       0.682
region_northwest    -0.0937      0.026     -3.536      0.000      -0.146      -0.042
region_southeast    -0.1043      0.027     -3.810      0.000      -0.158      -0.051
region_southwest    -0.0998      0.027     -3.700      0.000      -0.153      -0.047
==============================================================================
Omnibus:                       61.587   Durbin-Watson:                   1.955
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               56.825
Skew:                           0.448   Prob(JB):                     4.58e-13
Kurtosis:                       2.535   Cond. No.                         206.
==============================================================================
```

Fitting ordinary least squares model with all the features: Leading Current Reactive Power(LeRP) has Greatest P-value which is insignificant

SIGNIFICANCE TEST

OLS Regression Results

Dep. Variable:	binary_charges	R-squared (uncentered):	0.567
Model:	OLS	Adj. R-squared (uncentered):	0.564
Method:	Least Squares	F-statistic:	231.8
Date:	Fri, 08 Nov 2024	Prob (F-statistic):	3.81e-189
Time:	15:10:40	Log-Likelihood:	-702.76
No. Observations:	1069	AIC:	1418.
Df Residuals:	1063	BIC:	1447.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
children	0.0476	0.011	4.308	0.000	0.026	0.069
sex_male	0.0979	0.027	3.692	0.000	0.046	0.150
smoker_yes	0.7173	0.035	20.579	0.000	0.649	0.786
region_northwest	0.2650	0.033	7.988	0.000	0.200	0.330
region_southeast	0.2085	0.032	6.427	0.000	0.145	0.272
region_southwest	0.2346	0.034	6.887	0.000	0.168	0.301

Omnibus:	426.700	Durbin-Watson:	1.879
Prob(Omnibus):	0.000	Jarque-Bera (JB):	129.622
Skew:	0.659	Prob(JB):	7.13e-29
Kurtosis:	1.917	Cond. No.	5.27

Removing Insignificant variables: After removing variable having greatest P-value (greater than 0.05)

MULTICOLLINEARITY CHECK

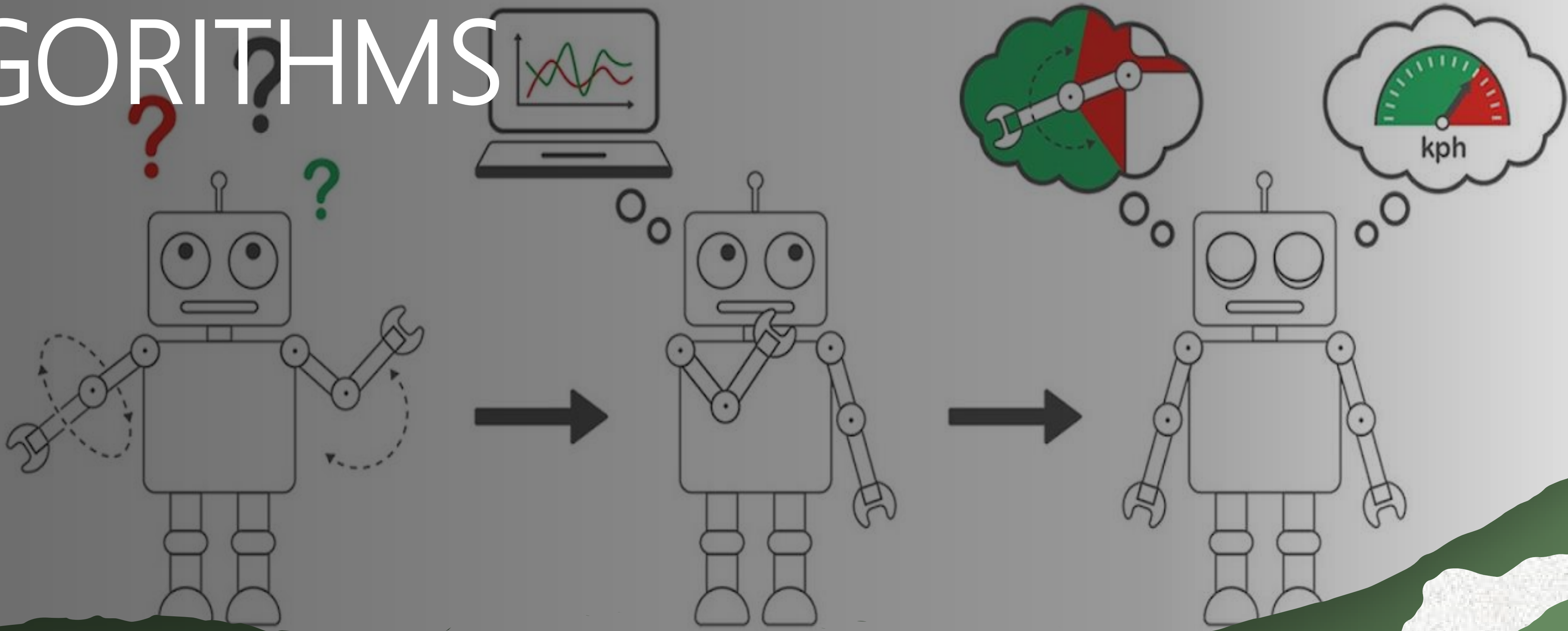
Variables with the greatest variance inflation factor (VIF >3) were removed

	variables	VIF
0	age	7.7
1	bmi	11.4
2	children	1.8
3	sex_male	2.0
4	smoker_yes	1.3
5	region_northwest	1.9
6	region_southeast	2.3
7	region_southwest	2.0

	variables	VIF
0	age	3.9
1	children	1.8
2	sex_male	1.9
3	smoker_yes	1.2
4	region_northwest	1.7
5	region_southeast	1.8
6	region_southwest	1.7

	variables	VIF
0	children	1.6
1	sex_male	1.7
2	smoker_yes	1.2
3	region_northwest	1.3
4	region_southeast	1.4
5	region_southwest	1.3

MACHINE LEARNING ALGORITHMS



MACHINE LEARNING ALGORITHMS

- Multiple Linear Regression
- K-Nearest Neighbors
- Decision Tree
- Random Forest
- Bagging
- Boosting
- Support Vector Machine

80:20 TRAIN-TEST SPLIT RATIO

Algorithms	Model 1	Model 2
Logistic Regression	0.914	0.675
KNN	0.865	0.634
Decision Tree	0.929	0.682
Random Forest	0.918	0.675
XG Boost	0.936	0.686
Ada Boost	0.891	0.682
SVM	0.921	0.682
ANN	0.932	0.716

***Model 1:** With all features* | ***Model 2:** After removing multi-collinear variables*

75:25 TRAIN-TEST SPLIT RATIO

Algorithms	Model 1	Model 2
Logistic Regression	0.916	0.677
KNN	0.850	0.629
Decision Tree	0.919	0.680
Random Forest	0.934	0.669
XG Boost	0.919	0.677
Ada Boost	0.910	0.686
SVM	0.925	0.680
ANN	0.933	0.723

Model 1: *With all features* | ***Model 2:*** *After removing multi-collinear variables*

70:30 TRAIN-TEST SPLIT RATIO

Algorithms	Model 1	Model 2
Logistic Regression	0.917	0.689
KNN	0.825	0.636
Decision Tree	0.925	0.694
Random Forest	0.943	0.692
XG Boost	0.932	0.706
Ada Boost	0.895	0.703
SVM	0.927	0.694
ANN	0.931	0.715

Model 1: *With all features |* ***Model 2:*** *After removing multi-collinear variables*

60:40 TRAIN-TEST SPLIT RATIO

Algorithms	Model 1	Model 2
Logistic Regression	0.908	0.693
KNN	0.814	0.639
Decision Tree	0.914	0.700
Random Forest	0.929	0.699
XG Boost	0.923	0.691
Ada Boost	0.917	0.700
SVM	0.917	0.697
ANN	0.938	0.722

Model 1: *With all features* | ***Model 2:*** *After removing multi-collinear variables*

ALGORITHMS COMPARISON (70:30)

Algorithms	Model 1	Model 2
Logistic Regression	0.917	0.689
KNN	0.825	0.636
Decision Tree	0.925	0.694
Random Forest	0.943	0.692
XG Boost	0.932	0.706
Ada Boost	0.895	0.703
SVM	0.927	0.694
ANN	0.931	0.715

SUMMARY

- For Model 1, the Random Forest algorithm provided the best results, demonstrating high accuracy and reliable predictions for structured data. It effectively handled the dataset's complexity and produced robust outcomes. In contrast, for Model 2, the Artificial Neural Network (ANN) outperformed other algorithms, showcasing its ability to capture intricate patterns and relationships in the data. This highlights the suitability of Random Forest for structured datasets and the strength of ANN in handling more complex, non-linear problems.

- These results highlight the importance of selecting the appropriate algorithm based on the characteristics of the dataset. Random Forest demonstrated superior performance in handling structured data, while ANN excelled in scenarios requiring advanced pattern recognition and non-linear analysis.

FUTURE SCOPE

- **Hybrid Models**: Combining Random Forest and ANN can leverage their strengths for improved predictions.

- **Optimization**: Advanced hyperparameter tuning methods like Grid Search or Bayesian Optimization can enhance performance.

- **Feature Engineering**: Using automated techniques can improve model accuracy and interpretability.

- **Diverse Applications**: The approach can be applied to complex datasets in healthcare, finance, and more.

- **Deep Learning Exploration**: Exploring architectures like CNNs or RNNs can uncover new possibilities.

- **Real-World Deployment**: Scaling and deploying models efficiently for practical use.

WORK DISTRIBUTION

Team	Contribution
Lokesh Kumar	Collect Information & Literature review
K.Manohar	Data Preprocessing
A.Deepak Mudiraj	Exploratory Data Analysis
N.Keerthana Reddy	Implement Machine Learning Algorithms

THANK YOU

Group 5
N.Keerthana Reddy
A.Deepak Mudiraj
Lokesh Kumar
K.Manohar

APPENDIX

LOADING THE DATASET

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

```
[ ] df.head()
```

	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

NULL VALUES

```
df.isna().sum()
```

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

dtype: int64

CHECKING FOR THE DATA TYPE

```
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
```

DEPENDENT VARIABLE

```
# Define the threshold value
threshold_value = 9386.1613

# Convert 'charges' column based on the threshold
df['binary_charges'] = np.where(df['charges'] <= threshold_value, 0, 1)

# Print the updated DataFrame
print(df[['charges', 'binary_charges']])
```

```
          charges  binary_charges
0     16884.92400               1
1      1725.55230               0
2      4449.46200               0
3     21984.47061               1
4      3866.85520               0
...           ...             ...
1333  10600.54830               1
1334   2205.98080               0
1335   1629.83350               0
1336   2007.94500               0
1337  29141.36030               1

[1337 rows x 2 columns]
```

DROPPING THE COLUMNS

```
[ ] X = df.drop(["charges", "binary_charges"], axis=1)
    y = df["binary_charges"]
```

```
[ ] # Count the occurrences of 0s and 1s in the binary_charges column
    counts = df['binary_charges'].value_counts()

    # Print the counts
    print(counts)
```

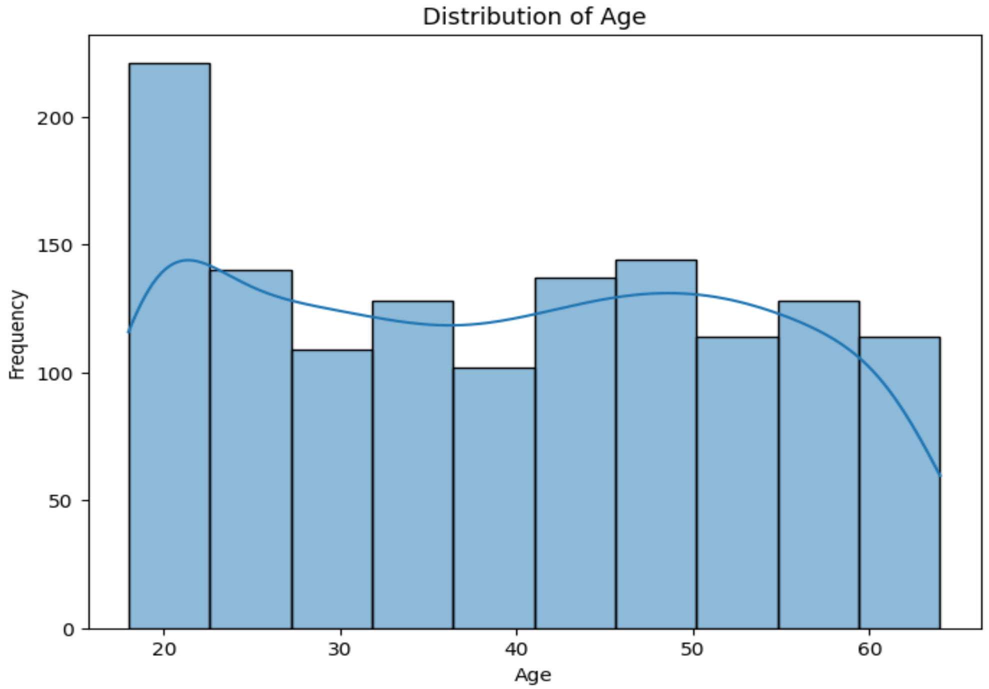
```
binary_charges
0    669
1    668
Name: count, dtype: int64
```

```
# Convert boolean columns to integers
bool_columns = X.select_dtypes(include=['bool']).columns
X[bool_columns] = X[bool_columns].astype(int)
X=pd.get_dummies(X,drop_first=True)
print(X)
```

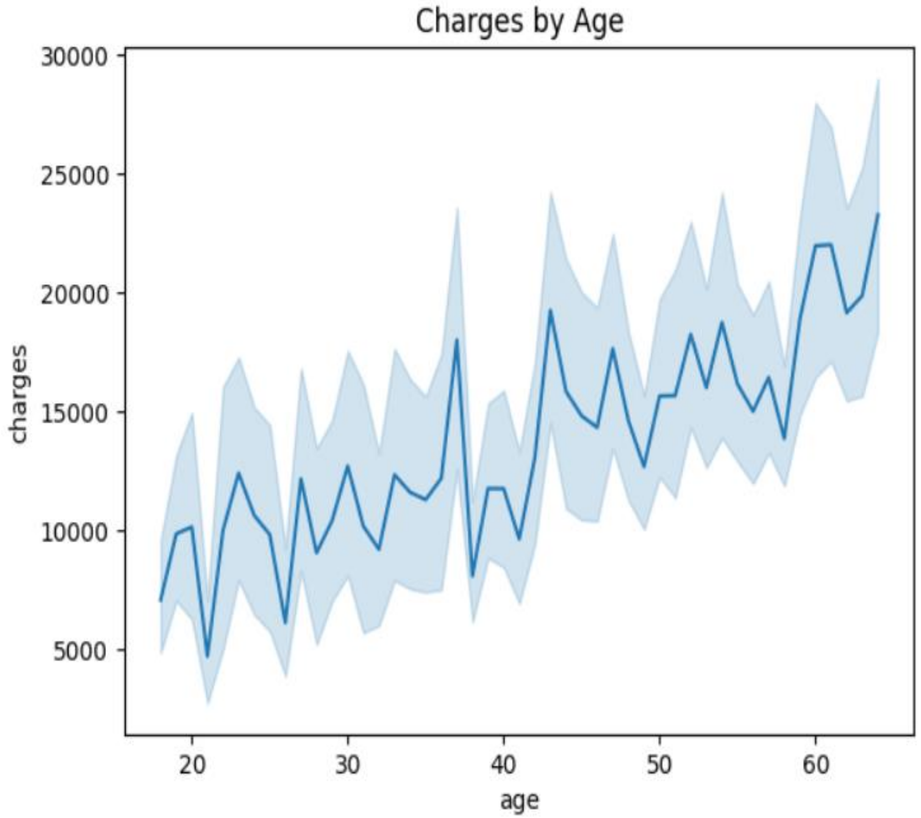
```
      age     bmi  children  sex_male  smoker_yes  region_northwest  \
0      19  27.900         0         0           1                 0
1      18  33.770         1         1           0                 0
2      28  33.000         3         1           0                 0
3      33  22.705         0         1           0                 1
4      32  28.880         0         1           0                 1
...   ...     ...       ...       ...         ...               ...
1333   50  30.970         3         1           0                 1
1334   18  31.920         0         0           0                 0
1335   18  36.850         0         0           0                 0
1336   21  25.800         0         0           0                 0
1337   61  29.070         0         0           1                 1

      region_southeast  region_southwest
0                    0                 1
1                    1                 0
2                    1                 0
3                    0                 0
4                    0                 0
...                ...               ...
1333                 0                 0
1334                 0                 0
1335                 1                 0
1336                 0                 1
1337                 0                 0

[1337 rows x 8 columns]
```

DATA VISUALIZATION

```python
plt.figure(figsize=(8, 6))
sns.histplot(df['age'], bins=10, kde=True)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```




```python
sns.lineplot(x='age', y='charges', data=df)
plt.title('Charges by Age')
plt.show()
```



MULTI COLLINEARITY CHECK

Multicollinearity

```
import warnings
warnings.filterwarnings("ignore")
```

```
# Split Data into Training and Test Sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
# Import library for VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(X):

    # Calculating VIF
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i).round(1) for i in range(X.shape[1])]

    return(vif)

calc_vif(X)
```

	variables	VIF
0	age	7.7
1	bmi	11.4
2	children	1.8
3	sex_male	2.0
4	smoker_yes	1.3
5	region_northwest	1.9
6	region_southeast	2.3
7	region_southwest	2.0

ORDINARY LEAST SQUARES

```python
import statsmodels.api as sm

model = sm.OLS(y, X).fit()

# Print the summary to get p-values
print(model.summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:          binary_charges   R-squared (uncentered):                   0.760
Model:                             OLS   Adj. R-squared (uncentered):              0.759
Method:                  Least Squares   F-statistic:                              526.8
Date:                 Fri, 08 Nov 2024   Prob (F-statistic):                        0.00
Time:                         15:10:40   Log-Likelihood:                         -478.48
No. Observations:                 1337   AIC:                                      973.0
Df Residuals:                     1329   BIC:                                      1015.
Df Model:                            8
Covariance Type:             nonrobust
====================================================================================
                       coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------
age                  0.0166      0.001     26.281      0.000       0.015       0.018
bmi                 -0.0052      0.001     -5.057      0.000      -0.007      -0.003
children            -0.0090      0.008     -1.152      0.249      -0.024       0.006
sex_male            -0.0533      0.019     -2.819      0.005      -0.090      -0.016
smoker_yes           0.6356      0.024     26.985      0.000       0.589       0.682
region_northwest    -0.0937      0.026     -3.536      0.000      -0.146      -0.042
region_southeast    -0.1043      0.027     -3.810      0.000      -0.158      -0.051
region_southwest    -0.0998      0.027     -3.700      0.000      -0.153      -0.047
==============================================================================
Omnibus:                       61.587   Durbin-Watson:                   1.955
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               56.825
Skew:                           0.448   Prob(JB):                     4.58e-13
Kurtosis:                       2.535   Cond. No.                         206.
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

LOGISTIC REGRESSION

75-25

```
logreg = LogisticRegression(C=1e9)

logreg.fit(X_train2_nomulti, y_train2_nomulti)
predictions2 = logreg.predict(X_test2_nomulti)
print(predictions2)
```

Show hidden output

```
[ ] z=confusion_matrix(y_test2_nomulti, predictions2)
    z
    accuracy_score(y_test2_nomulti,predictions2)
```

```
0.7134328358208956
```

```
[ ] print(classification_report(y_test2_nomulti,predictions2))
```

```
              precision    recall  f1-score   support

           0       0.64      0.98      0.77       168
           1       0.96      0.44      0.61       167

    accuracy                           0.71       335
   macro avg       0.80      0.71      0.69       335
weighted avg       0.80      0.71      0.69       335
```

K-NEAREST NEIGHBORS

```
[ ] from sklearn.neighbors import KNeighborsClassifier
```

```
[ ] model=KNeighborsClassifier(n_neighbors=5)
```

```
knn4 = pd.DataFrame({'Predicted':y_pred4,'Actual':y_test4_nomulti})
knn4
```

	Predicted	Actual
629	1	1
1087	0	1
283	1	1
790	0	0
594	0	0
...	...	...
479	0	0
538	0	0
117	1	1
5	0	0
1250	1	1

535 rows × 2 columns

```
[ ] from sklearn.metrics import accuracy_score
    accuracy_score(y_test4_nomulti,y_pred4)
```

```
0.6392523364485981
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
[ ] model=KNeighborsClassifier(n_neighbors=5)
```

80-20

```
[ ] model.fit(X_train1, y_train1)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
[ ] y_pred1 = model.predict(X_test1)
    y_pred1
```

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

DECISION TREE

```
[ ] # Create Decision Tree classifer object
    clf1 = DecisionTreeClassifier()

    # Train Decision Tree Classifer
    clf1 = clf1.fit(X_train1,y_train1)
```

```
[ ] DecisionTreeClassifier?
```

```
[ ] #Predict the response for test dataset
    y_pred1 = clf1.predict(X_test1)
```

```
[ ] # Model Accuracy, how often is the classifier correct?
    print("Accuracy:",metrics.accuracy_score(y_test1, y_pred1))

    Accuracy: 0.8880597014925373
```

```
# Create Decision Tree classifer object
clf3 = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf3 = clf3.fit(X_train3_nomulti,y_train3_nomulti)

#Predict the response for test dataset
y_pred3 = clf3.predict(X_test3_nomulti)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test3_nomulti, y_pred3))

Accuracy: 0.6940298507462687
```

SUPPORT VECTOR MACHINE

```
svm3 = pd.DataFrame({'Predicted':y_pred3,'Actual':y_test3_nomulti})
svm3
```

	Predicted	Actual
629	1	1
1087	0	1
283	0	1
790	0	0
594	0	0
...	...	...
924	0	0
873	0	0
731	0	1
363	0	0
10	0	0

402 rows × 2 columns

```
[ ] from sklearn.metrics import accuracy_score
    accuracy_score(y_test3_nomulti,y_pred3)
```

```
0.6940298507462687
```

```
[ ] from sklearn.svm import SVC
```

```
[ ] model = SVC(kernel='linear')
```

80-20

```
[ ] model.fit(X_train1, y_train1)
```

```
SVC
SVC(kernel='linear')
```

```
[ ] y_pred1 = model.predict(X_test1)
    y_pred1
```

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

RANDOM FOREST

```
[ ]  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
```

80-20

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

```
[ ]  rf.fit(X_train1_nomulti, y_train1_nomulti)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
[ ]  y_pred1 = rf.predict(X_test1_nomulti)
```

```
print(classification_report(y_test1, y_pred1))
print(confusion_matrix(y_test1_nomulti, y_pred1))
```

```
              precision    recall  f1-score   support

           0       0.63      0.90      0.74       145
           1       0.77      0.37      0.50       123

    accuracy                           0.66       268
   macro avg       0.70      0.64      0.62       268
weighted avg       0.69      0.66      0.63       268

[[131  14]
 [ 77  46]]
```

XGBOOST

```
[ ] import xgboost as xgb

    #Train the XGboost Model for Classification
    model1 = xgb.XGBClassifier()
    model2 = xgb.XGBClassifier(n_estimators=100, max_depth=8, learning_rate=0.1, subsample=0.5)
    model3 = xgb.XGBClassifier(n_estimators=100, max_depth=6, learning_rate=0.2, subsample=0.5)
```

80-20

```
[ ] train_model1 = model1.fit(X_train1_nomulti, y_train1_nomulti)
    train_model2 = model2.fit(X_train1_nomulti, y_train1_nomulti)
```

```
#prediction and Classification Report
from sklearn.metrics import classification_report

pred1 = train_model1.predict(X_test1_nomulti)
pred2 = train_model2.predict(X_test1_nomulti)

print('Model 1 XGboost Report %r' % (classification_report(y_test1_nomulti, pred1)))
print('Model 2 XGboost Report %r' % (classification_report(y_test1_nomulti, pred2)))
```

```
Model 1 XGboost Report '              precision    recall  f1-score   support\n\n           0       0.65      0.92      0.76       145\n           1       0.82      0.41      0.54
Model 2 XGboost Report '              precision    recall  f1-score   support\n\n           0       0.65      0.95      0.77       145\n           1       0.87      0.39      0.54
```

```
[ ] #Let's use accuracy score
    from sklearn.metrics import accuracy_score

    print("Accuracy for model 1: %.2f" % (accuracy_score(y_test1_nomulti, pred1) * 100))
    print("Accuracy for model 2: %.2f" % (accuracy_score(y_test1_nomulti, pred2) * 100))
```

```
Accuracy for model 1: 68.66
Accuracy for model 2: 69.40
```

ADA BOOST

```
[ ] from sklearn.ensemble import AdaBoostClassifier
```

```
[ ] # Initialize base estimator
    base_estimator = DecisionTreeClassifier(max_depth=3)  # Shallow trees typically work well with AdaBoost

    # Initialize AdaBoost model with the base estimator
    adaboost = AdaBoostClassifier(estimator=base_estimator, n_estimators=50, learning_rate=1.0, random_state=42)
```

80-20

```
[ ] # Train the model on the training data
    adaboost.fit(X_train1_nomulti, y_train1_nomulti)

    # Predict on the test set
    y_pred1 = adaboost.predict(X_test1_nomulti)

    # Calculate accuracy
    accuracy = accuracy_score(y_test1_nomulti, y_pred1)
    print("Accuracy:", accuracy)
```

```
Accuracy: 0.6828358208955224
```