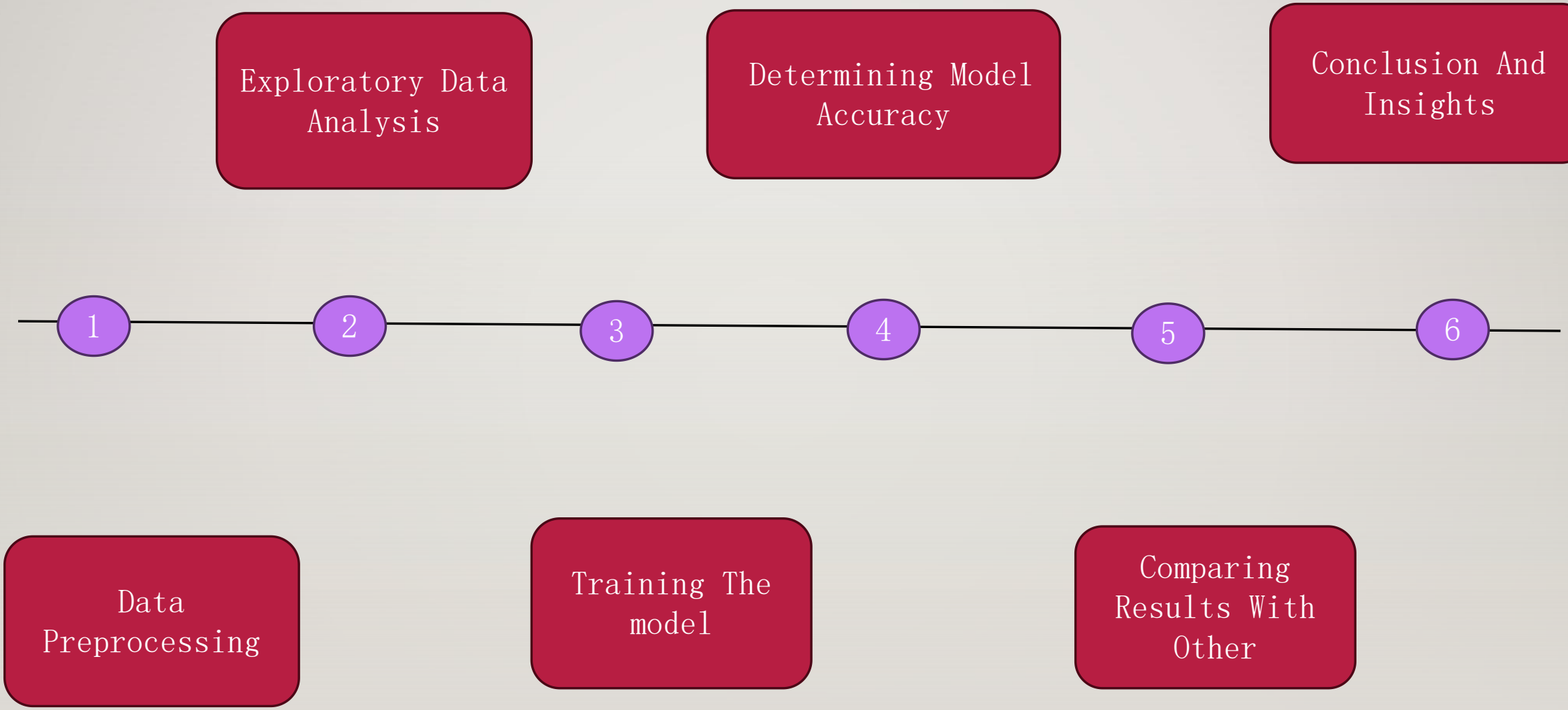
MEDICAL INSURANCE PREMIUM

TEAM
CHAVI LAHOTI
K.NIKHITHA

REGRESSION METHOD

BHARATHI DEGREE COLLEGE


Exploratory Data Analysis
Determining Model Accuracy
Conclusion And Insights
1
2
3
4
5
6
Data Preprocessing
Training The model
Comparing Results With Other

About The Data

- This insurance dataset contains information about individuals relevant to their medical insurance costs.It includes both demographic and lifestyle variables that are known to influence insurance charges.

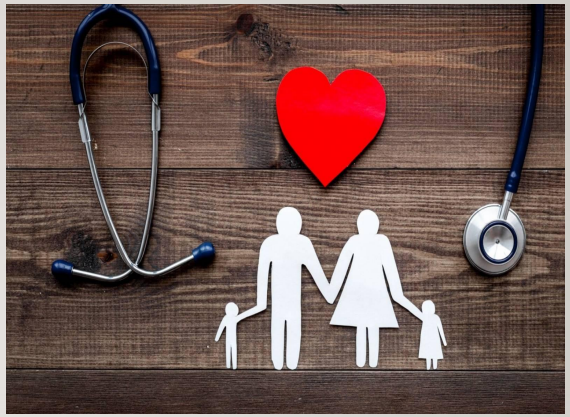
Objective

- The objective of this dataset is to model and predict the medical insurance charges for individual . By doing so, we aim to identify key cost drivers and assist providers in developing more precise and equitable pricing models.

The Path

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

Data And Data Quality



- Data Introduction:

The data consists of 7 columns and 1338 observations.

- Variables:

column	Description
Age	Age of the primary beneficiary (integer)
Sex	Gender of the individual (male / female)
Bmi	Body Mass Index - a measure of body fat (float)
Children	Number of children covered by health insurance (integer)
Smoker	Smoking status (yes / no)
Region	Residential region in the U.S. (northeast, etc.)
Charges	Individual medical costs billed by

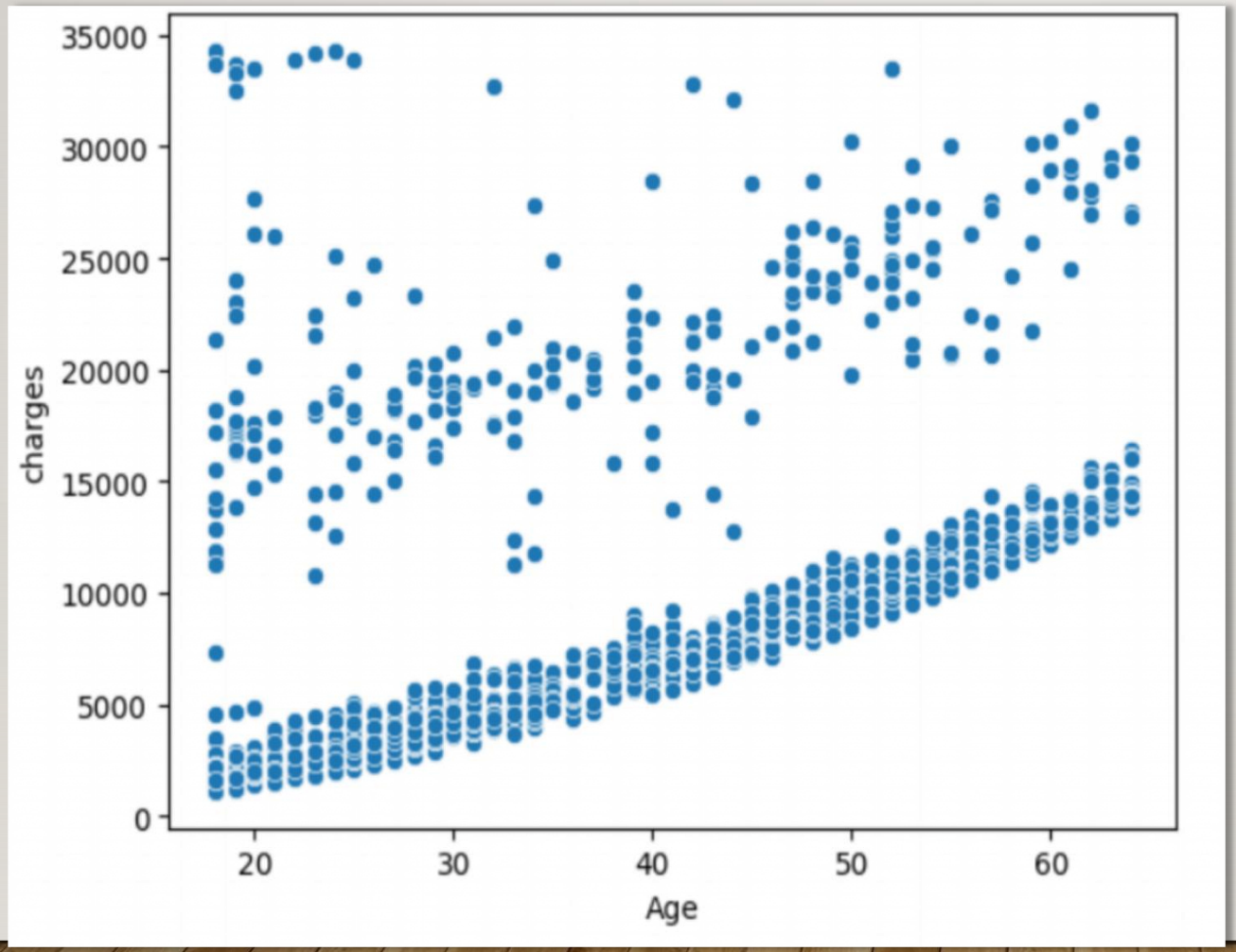
Missing Values

- There are no missing values in the data set. In regression we predict missing values based on other variables.

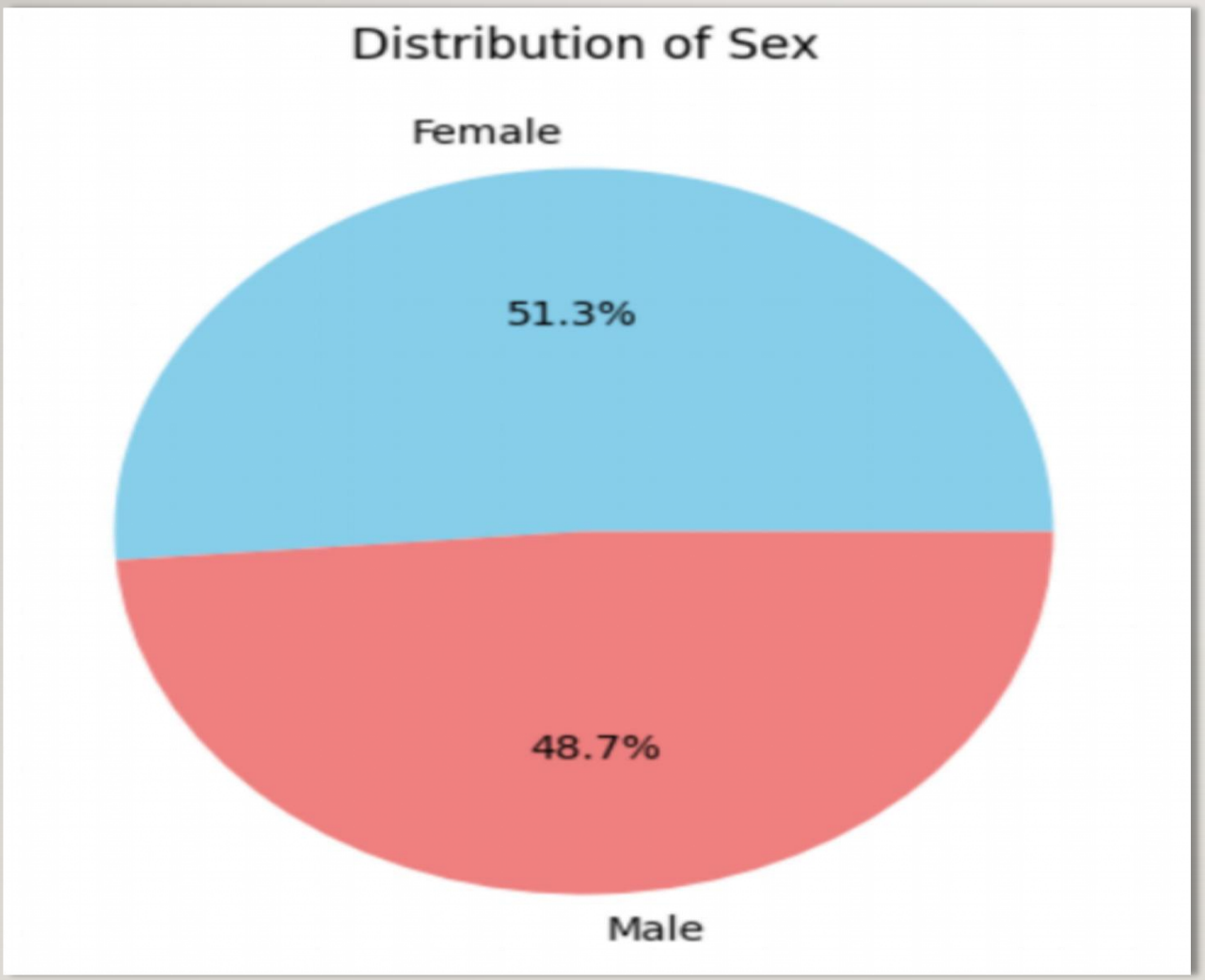
Dropped Columns

- Region Northeast column is been dropped in this data

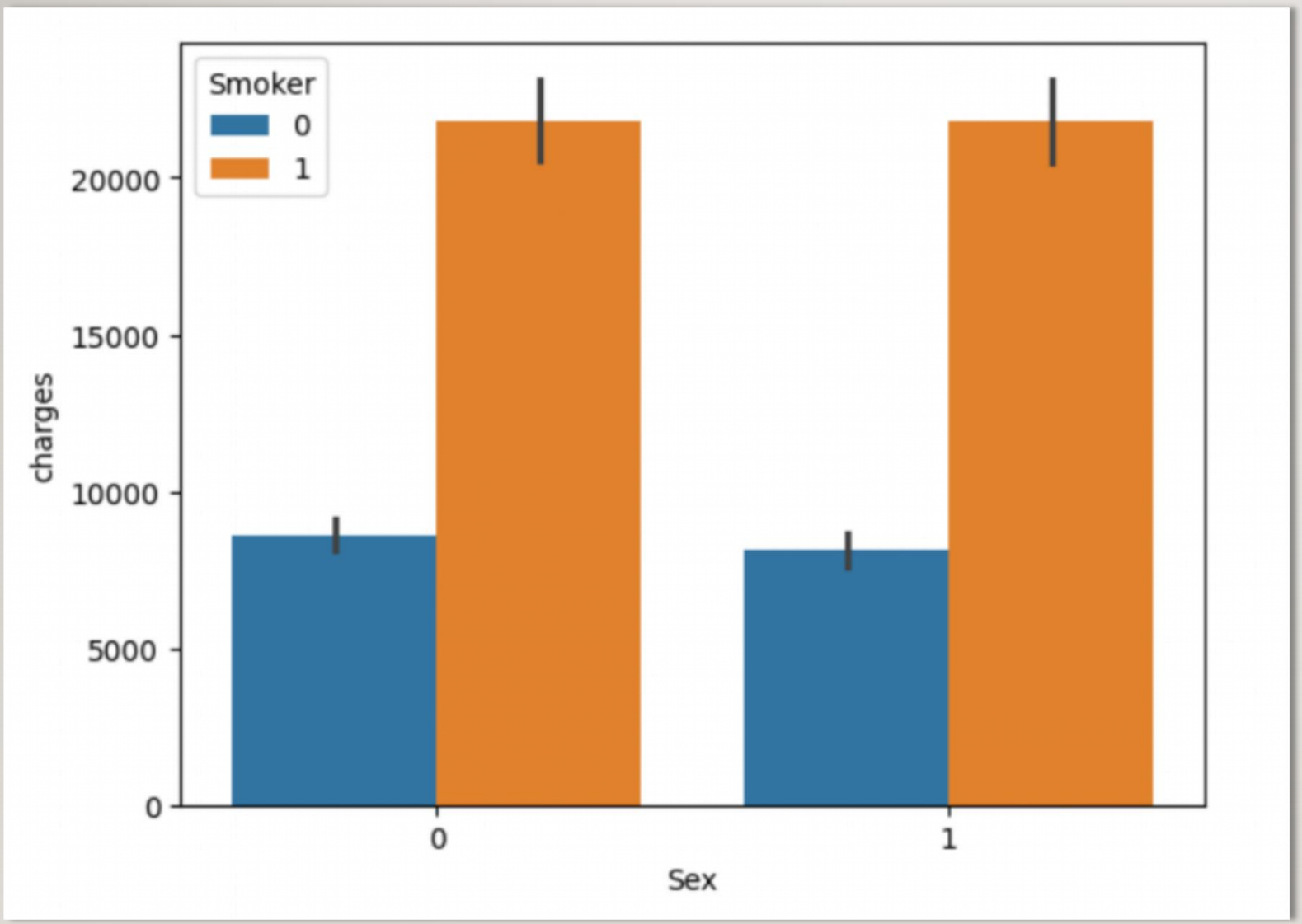
Data Visualization



Scatter plot of Age shows a positive correlation with Charges. As Age increases, the insurance charges also tend to increase.



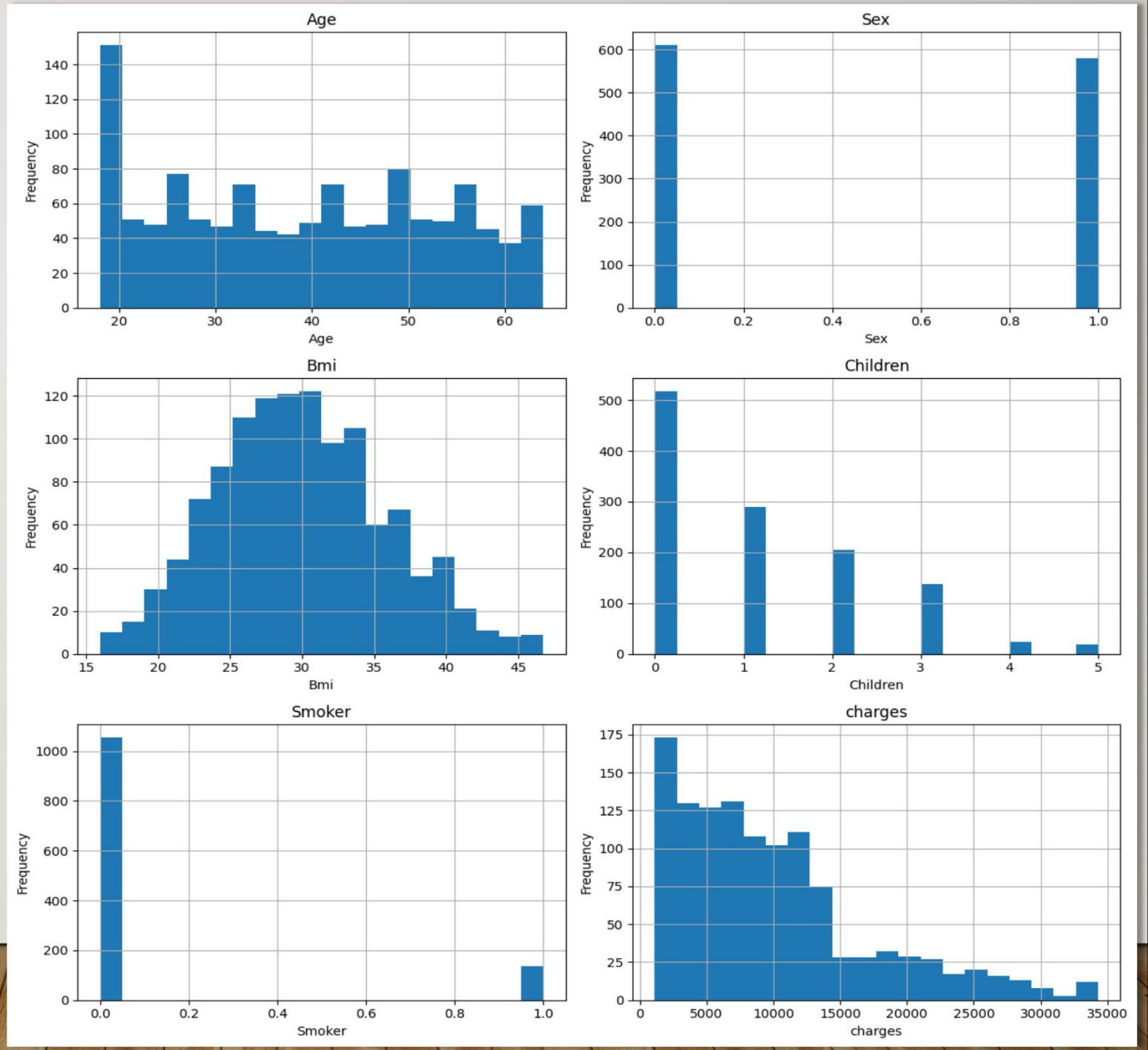
Pie chart shows distribution of sex with female as the majority at 51.3% and Male at 48.7% of the group.

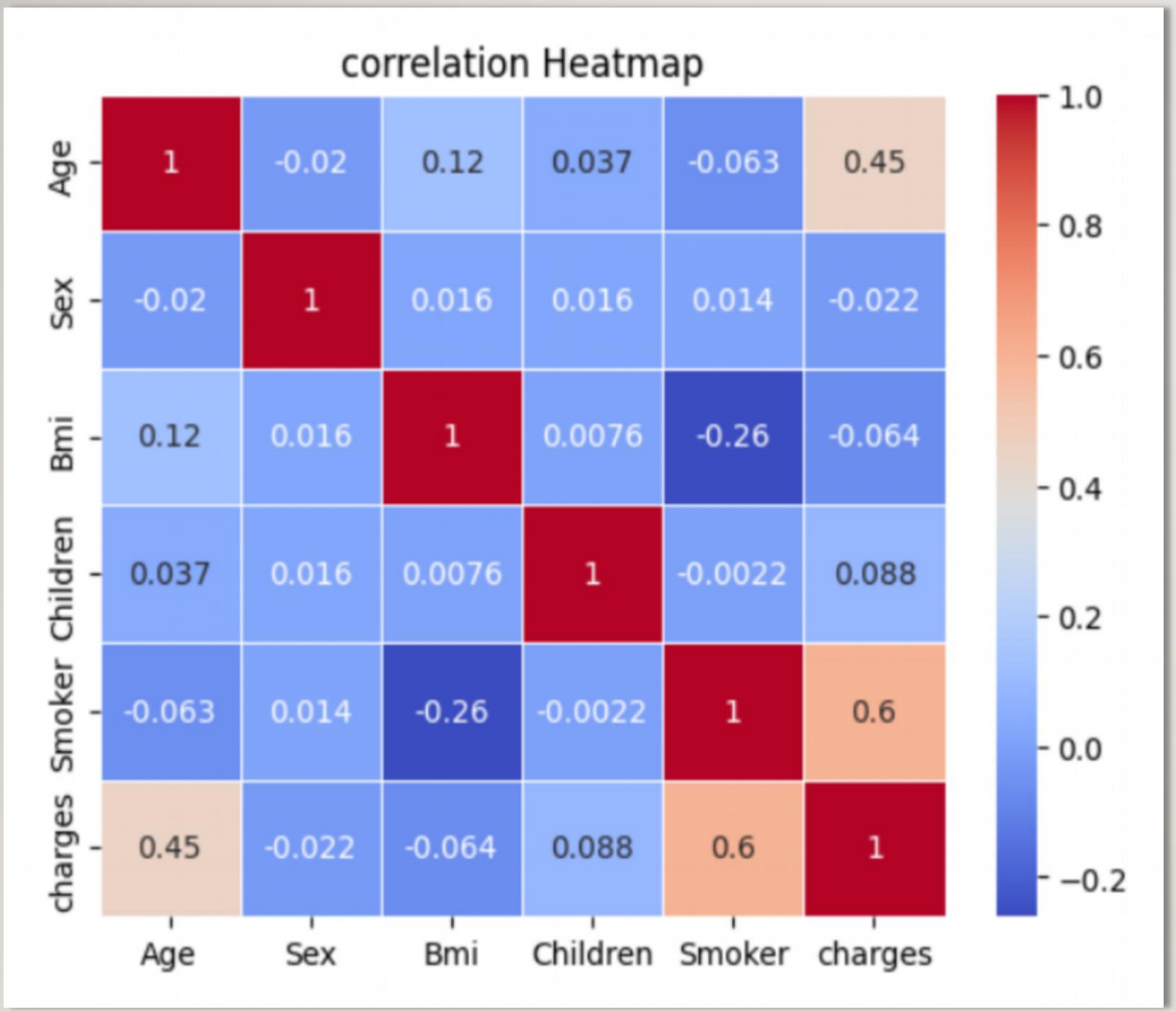


The bar plot shows that smoking has strong impact on insurance charges, far more than gender. Both male and female smokers incur significantly higher costs than non-smoker, highlighting smoking as a critical risk factor for insurers.

Histogram

These Histograms Shows the distribution of key variables in the insurance dataset.
Age, BMI, Children, Charges are right skewed.
Sex, Smoker are not skewed.





The correlation heatmap reveals that Smoking and Age are the two factors mostly positively correlated with insurance charges. Other variables like Sex, BMI and Children show weak or negligible correlation with charges.

Algorithms

Linear Regression

In conclusion, we get that 75-25 split provides the most optimal trade-off for linear regression model.

Train Test Ratio	R sq Value	MAE
80-20	0.697	4245.9
75-25	0.733	4018.4
70-30	0.740	4139.9
65-35	0.741	4162.3

KNN Regression

In conclusion, we get that 80-20 split provides the best balance between performance metrics.

Train Test Ratio	R sq Value	MAE
80-20	0.179	7374.7
75-25	0.152	7569.3
70-30	0.140	7900.4
65-35	0.120	8027.2

SVM Regression

In conclusion, we get that 75-25 split is best among the options for this SVR model.

Train Test Ratio	R sq Value	MAE
80-20	-0.153	6316.8
75-25	-0.088	5894.9
70-30	-0.123	6304.1
65-35	-0.169	6614.3

Decision Tree Regression

In conclusion, we get that 75-25 split provides the most optimal trade-off for the Decision Tree Regression model.

Train Test Ratio	MSE
80-20	152689971.1
75-25	146377534.3
70-30	159243811.9
65-35	171987642.6

XG Boost

In conclusion, we get that 75-25 split provides the best overall performance for the XG Boost model.

Train Test Ratio	R sq Value	MAE
80-20	0.780	3016.4
75-25	0.806	2911.2
70-30	0.813	2959.9
65-35	0.804	3109.3

ADA Boost

In conclusion, we get that 70-30 split provides the best performance in ADA Boost model.

Train Test Ratio	R sq Value	MAE
80-20	0.787	3804.9
75-25	0.801	3977.1
70-30	0.833	3565.7
65-35	0.803	4326.8

Model Comparison Table

Model	Mean Absolute Error(MAE)
Linear Regression	4018.4
KNN	7374.7
SVM	5894.9
XG Boost	2911.2
ADA Boost	3565.7

Conclusion

- In this medical insurance data set we have performed 6 machine learning algorithms.
- After applying these ML algorithms we conclude that XG Boost is best algorithm for 75-25 Train Test Ratio with Mean Absolute Error(MAE) of 2911.2 which is least error value among all.
- So XG Boost is best model for medical insurance premium

THANK YOU

Appendix

Training and Testing

```
[ ] from sklearn.model_selection import train_test_split

[ ] feature_cols = ['age','sex','bmi','children','smoker','region_northwest','region_southeast','region_southwest']
    X = data[feature_cols]
    Y = data["charges"]

[ ] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=1)
```

Linear Regression

```
[ ] import statsmodels.formula.api as smf
    from sklearn.linear_model import LinearRegression
    from sklearn import metrics
    from sklearn.model_selection import train_test_split
```

```
[ ] x = data[['Age','Sex','Bmi','Children','Smoker','Region_Northern West','Region_Southern East','Region_Southern West']]
    y = data[['charges']]
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=1)
    lm2 = LinearRegression()
    lm2.fit(X_train, Y_train)
    Y_pred = lm2.predict(X_test)
    print(np.sqrt(metrics.mean_squared_error(Y_test, Y_pred)))
```

```
[ ] from sklearn.metrics import r2_score
    r2_score(Y_test, Y_pred)
```

KNN

```
[ ] from sklearn.model_selection import train_test_split
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = .3, random_state = 46)

[ ] from sklearn.neighbors import KNeighborsRegressor

[ ] model=KNeighborsRegressor(n_neighbors=5)

[ ] model.fit(X_train, Y_train)
```

```
KNeighborsRegressor
KNeighborsRegressor()
```

```
[ ] Y_pred = model.predict(X_test)

    Y_pred
```

```
[ ] from sklearn.metrics import r2_score
    r2_score(Y_test, Y_pred)
```

```
0.17823615792544356
```

```
[ ] from sklearn import metrics
    metrics.mean_absolute_error(Y_test, Y_pred)
```

```
4518.356921567507
```

SVM

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

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

```
[ ] model.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of
  y = column_or_1d(y, warn=True)
```

SVR

SVR(kernel='linear')

```
[ ] Y_pred = model.predict(X_test)
```

```
[ ] Y_pred
```

```
[ ] from sklearn.metrics import r2_score
    r2_score(Y_test, Y_pred)
```

```
0.027875943830675376
```

```
[ ] from sklearn import metrics
    metrics.mean_absolute_error(Y_test,Y_pred)
```

```
3731.582514466458
```

Decision Tree

```
[ ]  import pandas as pd
     from sklearn.tree import DecisionTreeRegressor
     from sklearn.model_selection import train_test_split
     from sklearn import metrics
```

```
[ ]  dt = DecisionTreeRegressor()
```

```
[ ]  dt.fit(X_train, Y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor()
```

```
[ ]  reg = DecisionTreeRegressor()

     reg = reg.fit(X_train, Y_train)
```

```
[ ]  Y_pred = reg.predict(X_test)
```

```
[ ]  print(f"mean squared error:{mse:.4f}")
```

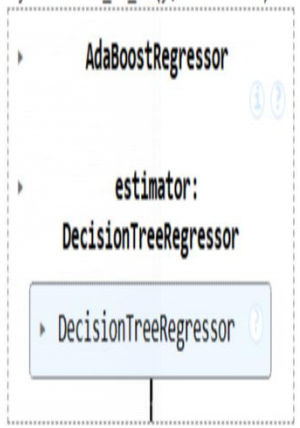
```
mean squared error:54471300.1112
```

XG Boost

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

```
[ ] base_estimator = DecisionTreeRegressor(max_depth=3, random_state=0)
    adaboost = AdaBoostRegressor(estimator=base_estimator, n_estimators=3, random_state=0)
    adaboost.fit(data, target)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vector y was passed
  y = column_or_1d(y, warn=True)
```




```
import xgboost as xgb # Import xgboost library
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error, mean_absolute_percentage_error
import numpy as np

# Define the base estimator
base_estimator = DecisionTreeRegressor(max_depth=3, random_state=0)

# Instantiate the AdaBoostRegressor model
xgboost = xgb.XGBRegressor(estimator=base_estimator, n_estimators=100, random_state=0)

# Fit the model to the training data
xgboost.fit(X_train, Y_train)

# Make predictions on the test data
Y_pred = xgboost.predict(X_test)

# Calculate evaluation metrics
mse = mean_squared_error(Y_test, Y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(Y_test, Y_pred)
mae = mean_absolute_error(Y_test, Y_pred)
mape = mean_absolute_percentage_error(Y_test, Y_pred) * 100

print(f"Mean Squared Error: {mse:.4f}")
print(f"Root Mean Squared Error: {rmse:.4f}")
print(f"R-squared: {r2:.4f}")
print(f"Mean Absolute Error: {mae:.4f}")
print(f"Mean Absolute Percentage Error: {mape:.4f}%")
```

ADA Boost

```
[ ] import xgboost as xgb
    from sklearn.model_selection import train_test_split
```

```
[ ] model1 = xgb.XGBRegressor()
    model2 = xgb.XGBRegressor(n_estimators=100, max_depth=8, learning_rate=0.1, subsample=0.5)

    train_model1 = model1.fit(X_train, Y_train)
    train_model2 = model2.fit(X_train, Y_train)
```

```
pred1 = train_model1.predict(X_test)
pred2 = train_model2.predict(X_test)
# Calculate and print the RMSE for both models
rmse1 = np.sqrt(mean_squared_error(Y_test, pred1))
rmse2 = np.sqrt(mean_squared_error(Y_test, pred2))

print(f'Model 1 XGBoost RMSE: {rmse1:.4f}')
print(f'Model 2 XGBoost RMSE: {rmse2:.4f}')
```

```
Model 1 XGBoost RMSE: 5315.3696
Model 2 XGBoost RMSE: 5127.5094
```

```
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error, mean_absolute_percentage_error
import numpy as np

# Define the base estimator
base_estimator = DecisionTreeRegressor(max_depth=3, random_state=0)

# Instantiate the AdaBoostRegressor model
adaboost = AdaBoostRegressor(estimator=base_estimator, n_estimators=100, random_state=0)

# Fit the model to the training data
adaboost.fit(X_train, Y_train)

# Make predictions on the test data
Y_pred = adaboost.predict(X_test)

# Calculate evaluation metrics
mse = mean_squared_error(Y_test, Y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(Y_test, Y_pred)
mae = mean_absolute_error(Y_test, Y_pred)
mape = mean_absolute_percentage_error(Y_test, Y_pred) * 100

print(f"Mean Squared Error: {mse:.4f}")
print(f"Root Mean Squared Error: {rmse:.4f}")
print(f"R-squared: {r2:.4f}")
print(f"Mean Absolute Error: {mae:.4f}")
print(f"Mean Absolute Percentage Error: {mape:.4f}%")
```

TEAM UNP

- AMAR SIR
- PALASH

SIR

- TEJA SIR