MAHATMA EDUCATION SOCIETY'S

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE
(Autonomous)

NEW PANVEL

PROJECT REPORT ON

"Analysing Patterns in Medical Cost"

IN PARTIAL FULFILLMENT OF

MASTER OF SCIENCE DATA ANALYTICS (PART I)

SEMESTER II– 2023-24

PROJECT GUIDE

Prof. Omkar Sherkhane

SUBMITTED BY: SHREYA BHATTACHARJEE

ROLL NO: 3108

Mahatma Education Society's

PILLAI COLLEGE OF ARTS,COMMERCE & SCIENCE (Autonomous)

Re-accredited “A” Grade by NAAC (3rd Cycle)

Project Completion Certificate

THIS IS TO CERTIFY THAT

SHREYA BHATTACHARJEE

of **M.Sc. Data Analytics Part – I** has completed the project titled ‘**Analysing Patterns in Medical Cost**’ of subject **Big Data Analytics** under our guidance and supervision during the academic year 2023-24 in the department of Computer Science.

INTRODUCTION

The dataset is often used in educational settings and is publicly available for research and analysis. The exact origin of the dataset may vary depending on where it's obtained from

Attributes: The dataset typically includes several attributes for each individual, such as:

- Age: The age of the individual.
- Sex: The gender of the individual.
- BMI (Body Mass Index): A measure of body fat based on height and weight.
- Number of Children: The number of children/dependents covered by the insurance plan.
- Smoking Status: Whether the individual is a smoker or non-smoker.
- Region: The geographic region of the individual (though this may be omitted or treated differently in some versions of the dataset).
- Medical Charges: The amount charged by the medical insurance for the individual's healthcare coverage.

1. **Purpose:** The dataset is commonly used for tasks such as predictive modelling, regression analysis, and exploring factors influencing medical costs. Researchers and analysts may use it to understand how various factors such as age, BMI, smoking habits, and geographic location impact medical expenses.

2. **Size:** The dataset typically contains 1338 Rows and 7 Column

3. **Format:** It is usually provided in a tabular format, such as a CSV (Comma Separated Values) file, with each row representing an individual and each column representing an attribute.

4. **Usage:** The dataset is widely used in educational contexts for teaching purposes, as well as in research and data analysis projects in the fields of healthcare, insurance, and machine learning.

5. **License:** The dataset may have different licensing terms depending on the source. In many cases, it's freely available for non-commercial and educational purposes.

TOOLS AND TECHNIQUES USED

Google Colab Notebook:

Google Colab is a free, cloud-based platform that allows you to write and execute Python code in a collaborative environment. It provides access to GPU and TPU accelerators, making it suitable for training machine learning models. Google Colab notebooks are frequently utilized due to their convenient cloud-based environment

I. **Pandas:** Pandas is a Python library used for data manipulation and analysis. It provides data structures like Data Frames and tools for reading and writing data between different formats.

II. **NumPy**: NumPy is a Python library used for numerical computing. It provides support for arrays, matrices, and mathematical functions to operate on these data structures efficiently.

III. **Matplotlib:** Matplotlib is a plotting library for Python used to create static, interactive, and animated visualizations in Python.

IV. **Seaborn:** Seaborn is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

V. **Scikit-learn (sklearn):** Scikit-learn is a machine learning library for Python. It provides simple and efficient tools for data mining and data analysis, including classification, regression, clustering, dimensionality reduction, and more.

VI. **StandardScaler:** StandardScaler is a pre-processing technique from Scikit-learn used for standardizing features by removing the mean and scaling to unit variance.

VII. **LabelEncoder:** LabelEncoder is a pre-processing technique from Scikit-learn used to convert categorical labels into numerical labels.

VIII. **Train-Test Split:** Train-Test Split is a technique used to split the dataset into two subsets: one for training the model and the other for evaluating its performance.

IX. **Linear Regression**: Linear Regression is a supervised learning algorithm used for modelling the relationship between one or more independent variables and a dependent variable.

X. **Support Vector Machine (SVM):** Support Vector Machine is a supervised learning algorithm used for regression tasks. It uses support vectors to approximate the function that maps input data points to the target values.

XI. **K-Nearest Neighbors (KNN):** K-Nearest Neighbors is a supervised learning algorithm used for classification and regression tasks. It predicts the target variable by averaging the values of its k nearest neighbors in the feature space.

XII. **Mean Squared Error (MSE):** Mean Squared Error is a metric used to evaluate the performance of regression models. It measures the average squared difference between the predicted values and the actual values.

PROJECT METHODOLOGY

1. Data Loading and Understanding:

Begin by loading the dataset containing information about insurance charges.
Examine the structure of the data, including the number of entries, data types, and any missing values.

2. Exploratory Data Analysis (EDA):

Perform EDA to gain insights into the distribution and relationships between different variables.
Utilize data visualization techniques such as histograms, boxplots, scatter plots, and heatmaps to visualize the data.

3. Data Pre-processing:

Prepare the data for modelling by handling any missing values and converting categorical variables into numerical representations.
Normalize numerical features to ensure uniform scaling and improve model performance.

4. Model Selection and Training:

Select appropriate machine learning models for regression analysis, considering the nature of the problem and the dataset.
Split the data into training and testing sets to evaluate model performance.
Train different regression models such as Linear Regression, Support Vector Regressor (SVR), and K-Nearest Neighbors (KNN) Regressor.

5. Model Evaluation:

Evaluate the trained models using appropriate regression metrics such as mean squared error (MSE) and R-squared.
Compare the performance of different models to identify the most suitable one for predicting insurance charges.

6. Visualization and Interpretation:

Visualize the performance of different models using bar charts or other graphical representations.

Interpret the results and provide insights into the factors influencing insurance charges based on the chosen model.

7. Conclusion

Summarize the findings from the analysis and highlight key insights.

Provide recommendations based on the analysis results, such as strategies to mitigate high insurance charges or target specific demographic groups for insurance offerings.

CODE

```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
# load the dataset
data = pd.read_csv('/content/insurance.csv')
```

```
data.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
```

```
data.shape
```

(1338, 7)

(1338, 7)

```
data.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

```
data.tail()
```

	age	sex	bmi	children	smoker	region	charges
1333	50	male	30.97	3	no	northwest	10600.5483
1334	18	female	31.92	0	no	northeast	2205.9808
1335	18	female	36.85	0	no	southeast	1629.8335
1336	21	female	25.80	0	no	southwest	2007.9450
1337	61	female	29.07	0	yes	northwest	29141.3603

```
data.describe()
```

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

```
data.dtypes
```

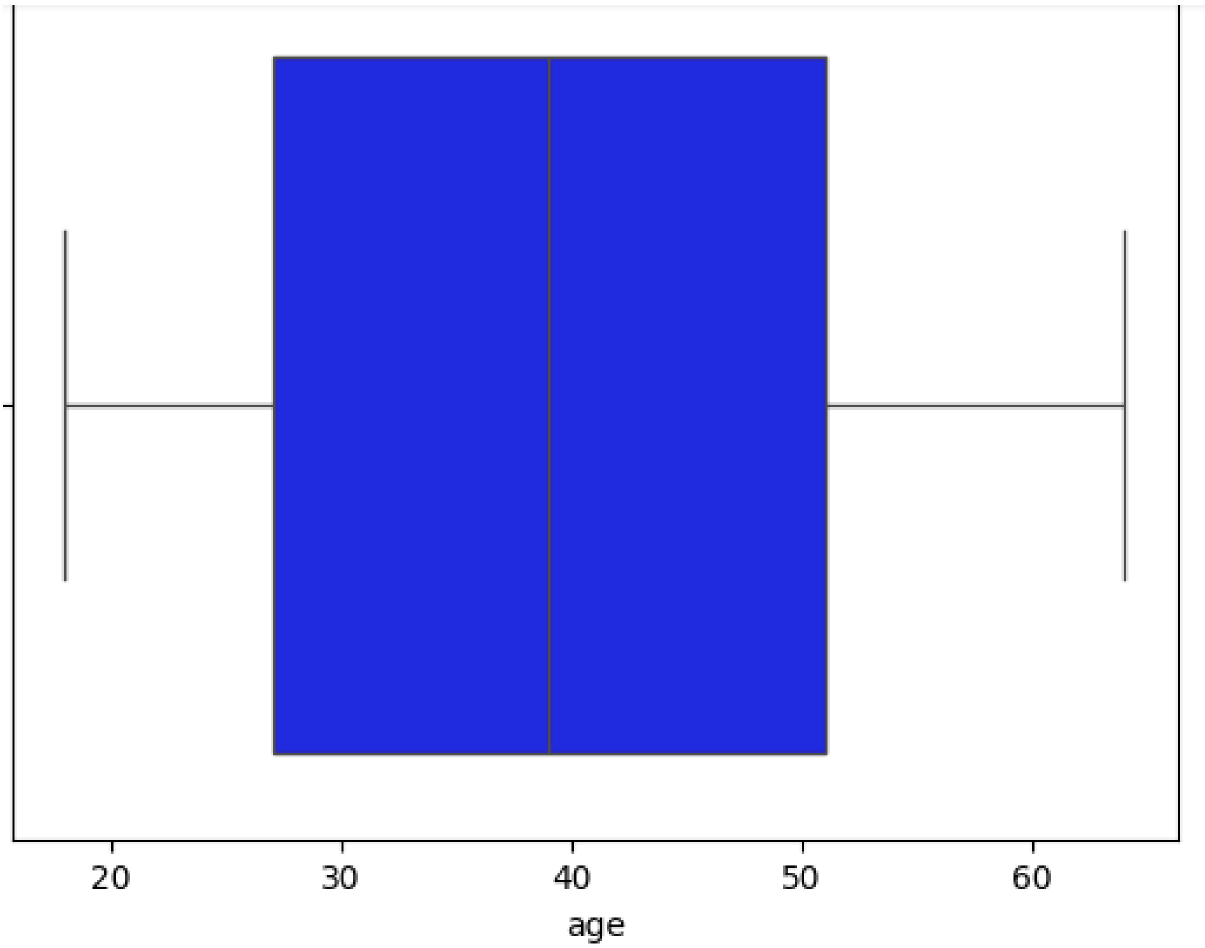
```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
charges     float64
dtype: object
```

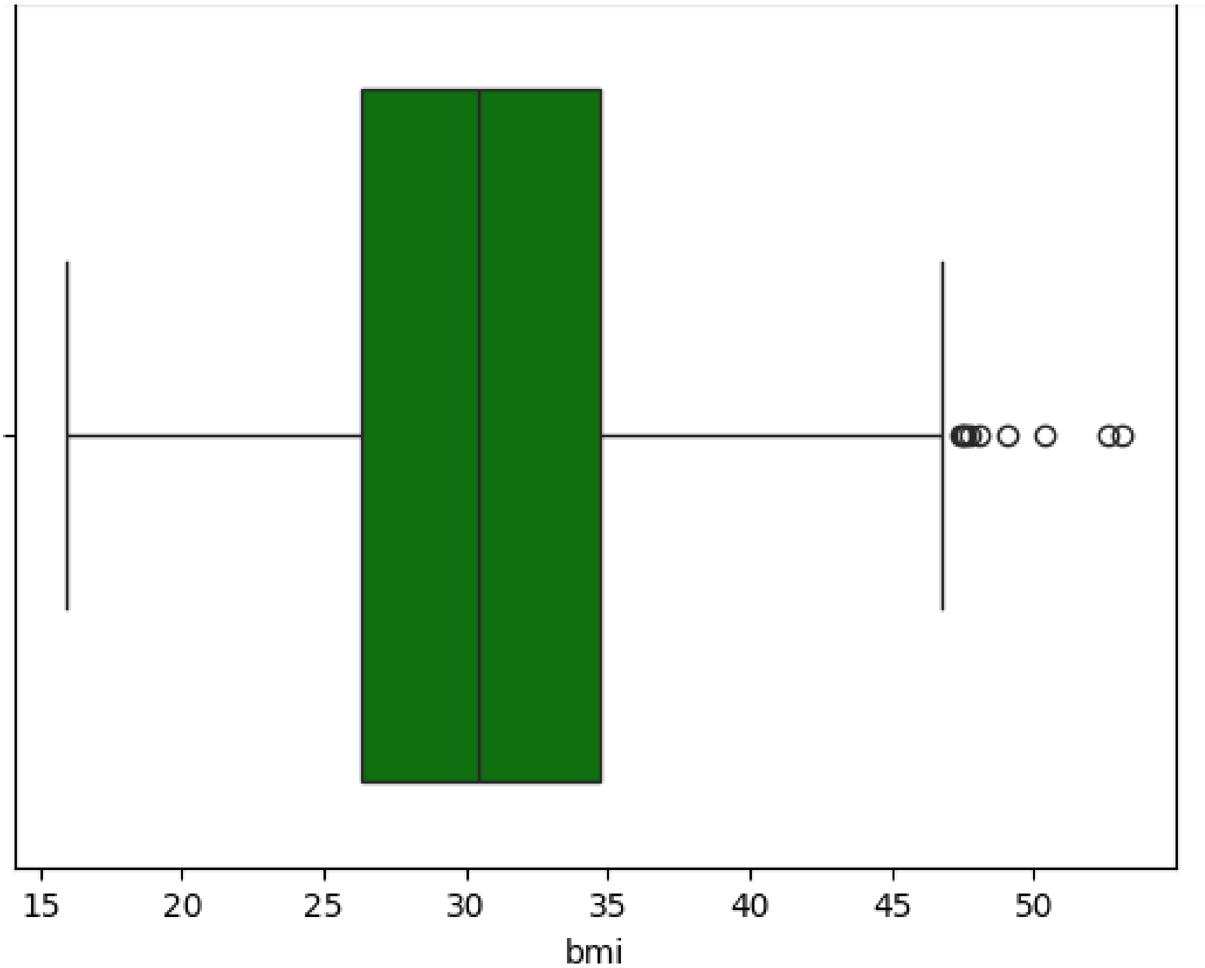
```
data.isnull().sum()
```

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

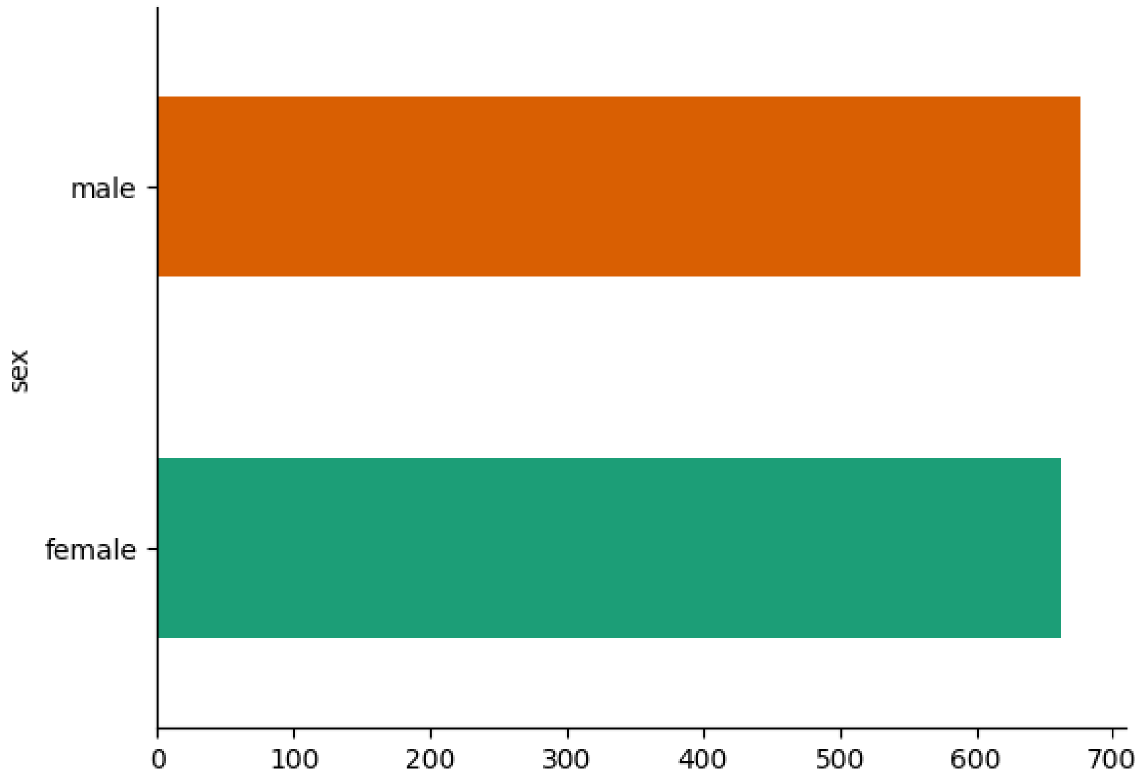
Visualization

```
column = ['age', 'bmi', 'children', 'charges']
colors = ['blue', 'green', 'orange', 'red']  # Define
colors for each variable

for i, col in enumerate(column):
    sns.boxplot(x=data_pre_pro[col], color=colors[i])
# Use color parameter with the specified color
    plt.xlabel(col)
    plt.show()
```

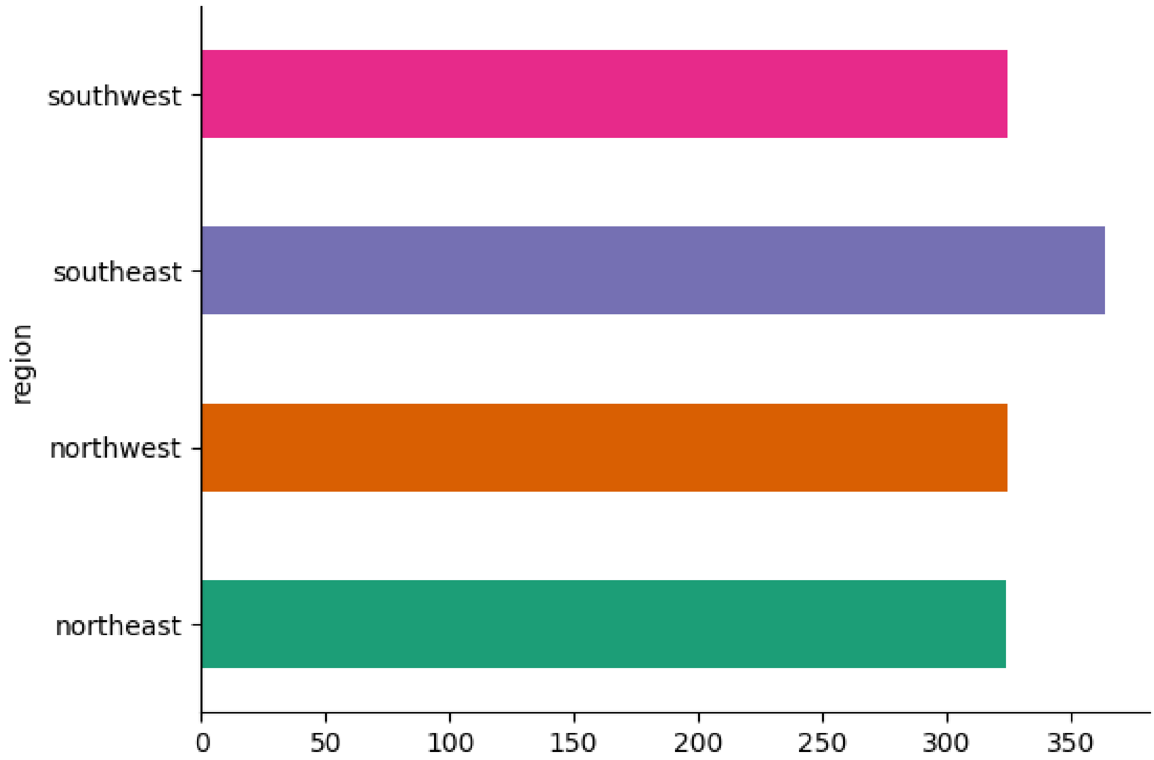




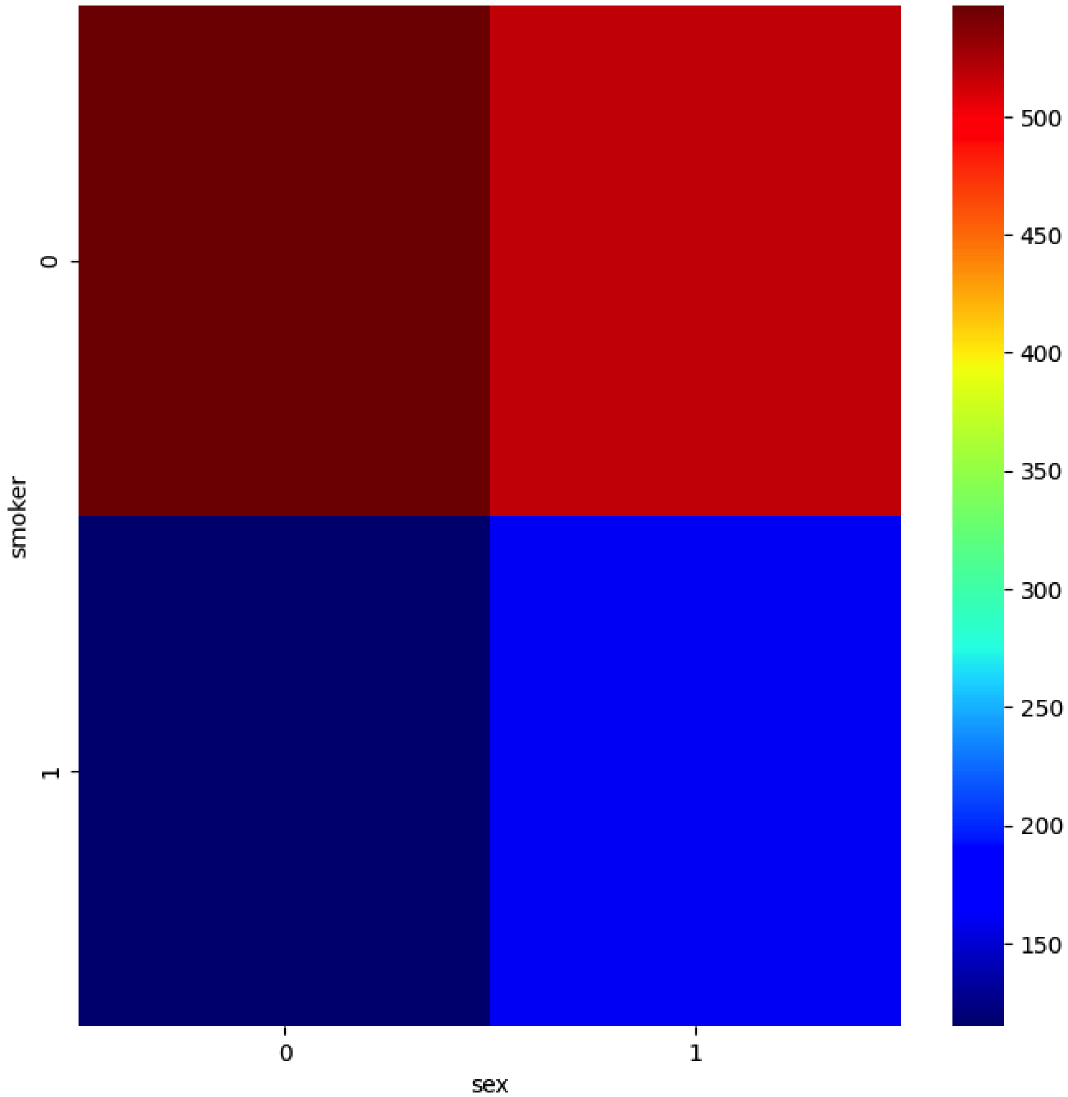

```
#sex

from matplotlib import pyplot as plt
import seaborn as sns
data.groupby('sex').size().plot(kind='barh',
color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```




```
# @title region

from matplotlib import pyplot as plt
import seaborn as sns
data.groupby('region').size().plot(kind='barh',
color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```




```
# @title sex vs smoker

from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['smoker'].value_counts()
    for x_label, grp in data.groupby('sex')
})
sns.heatmap(df_2dhist, cmap='jet')
plt.xlabel('sex')
_ = plt.ylabel('smoker')
```




```
# Converting objects labels into categorical
data[['sex', 'smoker', 'region']] = data[['sex',
'smoker', 'region']].astype('category')
data.dtypes
```

```
age            int64
sex         category
bmi          float64
children       int64
smoker      category
region      category
charges      float64
dtype: object
```

```python
# Converting category labels into numerical using LabelEncoder
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
label.fit(data.sex.drop_duplicates())
data.sex = label.transform(data.sex)
label.fit(data.smoker.drop_duplicates())
data.smoker = label.transform(data.smoker)
label.fit(data.region.drop_duplicates())
data.region = label.transform(data.region)
data.dtypes
```

```
age           int64
sex           int64
bmi         float64
children      int64
smoker        int64
region        int64
charges     float64
dtype: object
```

```python
f, ax = plt.subplots(1, 1, figsize=(10, 10))
ax = sns.heatmap(data.corr(), annot=True, cmap='viridis')
```


```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import  StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import learning_curve
from sklearn.metrics import mean_squared_error
```

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
```

```python
# Step 1: Handling Null Values
if data.isnull().sum().sum():
    data = data.dropna()

# Step 2: Handling String Values ('smoker' Column)
data['smoker'] =
data['smoker'].replace(to_replace=['no', 'yes'],
value=[0, 1])



# Step 3: Converting Categorical Variables to
Dummy/Indicator Variables
data = pd.get_dummies(data)

# Step 4: Normalization (Scaling)
scaler = StandardScaler()
data['charges'] =
scaler.fit_transform(data[['charges']])
data['bmi'] = scaler.fit_transform(data[['bmi']])
data['age'] = scaler.fit_transform(data[['age']])

# Resulting preprocessed DataFrame
data
```

	age	sex	bmi	children	smoker	charges
0	-1.438764	0	-0.453320	0	1	0.298584
1	-1.509965	1	0.509621	1	0	-0.953689
2	-0.797954	1	0.383307	3	0	-0.728675
3	-0.441948	1	-1.305531	0	0	0.719843
4	-0.513149	1	-0.292556	0	0	-0.776802
...	...	...	...	...	...	...
1333	0.768473	1	0.050297	3	0	-0.220551
1334	-1.509965	0	0.206139	0	0	-0.914002
1335	-1.509965	0	1.014878	0	0	-0.961596
1336	-1.296362	0	-0.797813	0	0	-0.930362
1337	1.551686	0	-0.261388	0	1	1.311053

1338 rows × 6 columns

```
X = data
y = data['charges']
X = X.drop("charges" , axis = 1)
```

```
# divide the data into train and test
X_train, X_test, y_train, y_test =
train_test_split(X,y,train_size=1000)
```

```
#First model is Linear Regression
model = LinearRegression() # define instance
model.fit(X_train, y_train) # passing the data and fit
it
score = model.score(X_test , y_test) # test
theprediction
score
```

> **0.7548178846952174**

```
# model_2 = SVR(kernel='poly') # define instance *kenel
= 'poly' to try find non linear separate

model_2 = SVR(kernel='rbf') # define instance *kenel =
'rbf' to try find non linear separate
model_2.fit(X_train, y_train) # passing the data and
fit it
score_2 = model_2.score(X_test , y_test) # test the
prediction
score_2
```

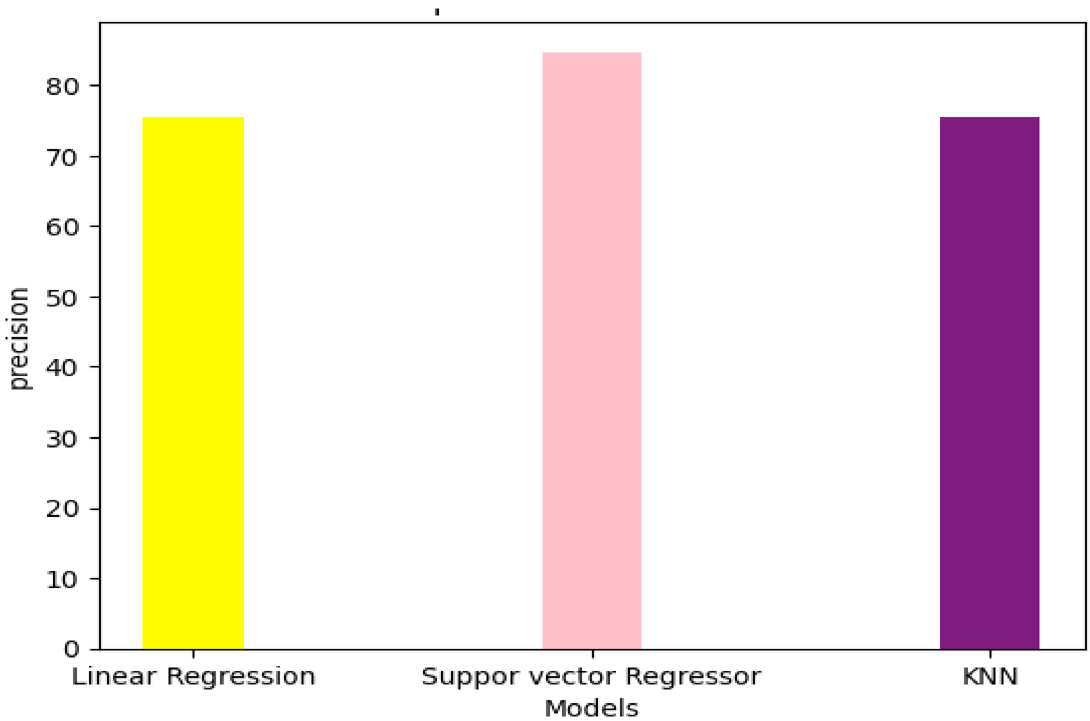
> **0.8469338128290789**

```
# model_3 = KNeighborsRegressor(n_neighbors= 5) #
define instance
model_3 = KNeighborsRegressor(n_neighbors= 3) # define
instance
```

```
model_3.fit(X_train, y_train) # passing the data and
fit it
score_3 = model_3.score(X_test,y_test) # test the
prediction
score_3
```

> **0.755437177892548**

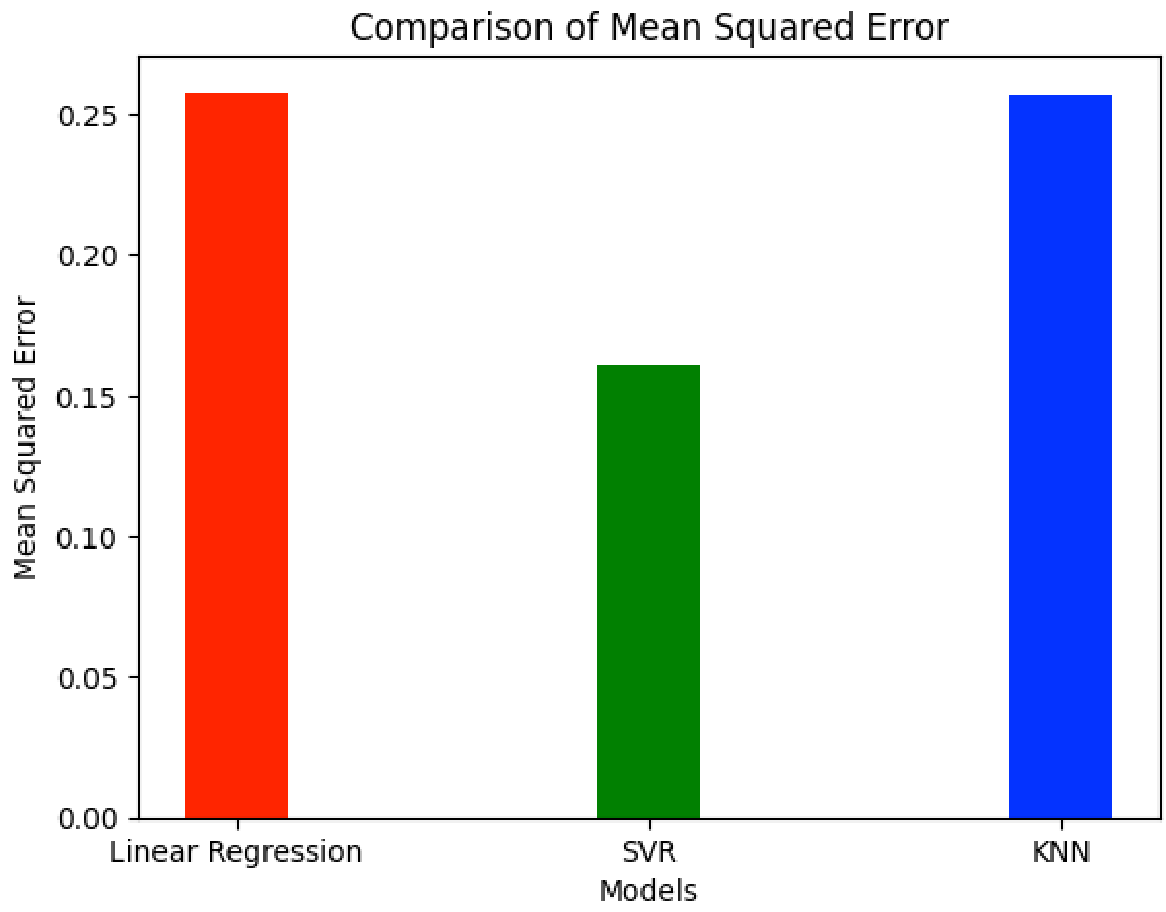
```
models = ['Linear Regression', 'Suppor vector
Regressor',  'KNN']

scores = [score * 100, score_2 * 100, score_3 * 100]

plt.bar(models, scores, color=['yellow', 'pink',
'purple'], width = 0.25)

plt.xlabel("Models")
plt.ylabel("precision")
plt.title("Comparison of Model Scores")
plt.show()
```



```python
# Linear Regression
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Linear Regression MSE: {mse}")

# SVR
model_2 = SVR(kernel='rbf')
model_2.fit(X_train, y_train)
y_pred_2 = model_2.predict(X_test)
mse_2 = mean_squared_error(y_test, y_pred_2)
print(f"SVR MSE: {mse_2}")

# KNN
model_3 = KNeighborsRegressor(n_neighbors= 3)
model_3.fit(X_train, y_train)
y_pred_3 = model_3.predict(X_test) # make predictions
mse_3 = mean_squared_error(y_test, y_pred_3)
print(f"KNN MSE: {mse_3}")
```

```
Linear Regression MSE: 0.2576619362782812
SVR MSE: 0.16085728812710032
KNN MSE: 0.2570112092761896
```

```python
# plot a graph
mse_values = [mse, mse_2, mse_3]

models = ['Linear Regression', 'SVR', 'KNN']

plt.bar(models, mse_values, color=['red', 'green',
'blue'], width = 0.25)
plt.xlabel('Models')
plt.ylabel('Mean Squared Error')
plt.title('Comparison of Mean Squared Error')
plt.show()
```



CONCLUSION

Our analysis provides insights into the factors influencing medical costs and demonstrates the effectiveness of machine learning models in predicting healthcare expenses. The SVR model, in particular, shows promise for accurately estimating medical charges based on patient characteristics. Overall, this project contributes to the understanding of medical cost patterns and offers valuable insights for healthcare providers, policymakers, and individuals seeking to optimize healthcare resource allocation and improve cost management strategies.