In Summary;

This project focuses on predicting insurance costs based on demographic and lifestyle factors such as age, sex, BMI, number of children, smoking habits, and more using Python. After importing the dataset, I performed an initial exploration and descriptive analysis to understand the data structure. The "charges" column, representing insurance costs, was separated as the target variable, while the remaining columns were used as independent features for prediction. Categorical values were transformed into nominal values to facilitate machine learning processes, and unnecessary columns were removed to streamline the dataset. The data was then split into training and testing sets to evaluate model performance.

To predict insurance charges, I tested various regression models, including decision tree regression, random forest regression, lasso regression, elastic net regression, and ridge regression. Among these, the random forest model emerged as the most accurate for this dataset. I further imported a new dataset for prediction purposes, preprocessing it by converting categorical values to nominal values and removing unnecessary columns. Using the trained model, I predicted the "charges" column for the new dataset and incorporated the predictions into the dataset. Finally, I converted numerical values back into object types for usability and saved the updated dataset for further analysis or reporting. This end-to-end process demonstrates the integration of exploratory data analysis, preprocessing, and machine learning to address practical insurance cost prediction challenges.

INSURANCE PREDICTION

* Import the necessary libraries.

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt


from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler


from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet


from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

* Import the data.

```
data1 =
pd.read_csv('C:\\Users\\Asus\\OneDrive\\Masaüstü\\website\\python\\PREDICTION\\healt
hcare_insurance_cost_prediction\\insurance.csv')

data1.head()
```

* Use 6 columns, to predict "charges" column.

* Transform string values to numeric values to make prediction. For prediction we need numeric values.

```
data1.describe()
```

* Min age 18 normal, min children 0 normal, there is no failure.

```
data1.info()

print(data1.isnull().sum())
```

* There is no null.

* See the total number of people by age.

```
age_stat = data1['age'].value_counts()
sex_stat = data1['sex'].value_counts()
bmi_stat = data1['bmi'].value_counts()
children_stat = data1['children'].value_counts()
smoker_stat = data1['smoker'].value_counts()
region_stat = data1['region'].value_counts()
print(age_stat)
```

* See the total number of people by sex.

```
print(sex_stat)
```

* See the total number of people by body mass index.

```
print(bmi_stat)
```

* See the total number of people by children.

```
print(children_stat)
```

* See the total number of people by smoking.

```
print(smoker_stat)
```

* See the total number of people by region.

```
print(region_stat)
```

* Dataset looks it hasnt any problem.

* Transform cathegorical values to nominal values.

* Transform male and female to 0 and 1. same for smoker and region too.

```
sex_types = pd.get_dummies(data1.sex, prefix = "sex")
sex_types = sex_types.astype(int)
smoker_types = pd.get_dummies(data1.smoker, prefix = "smoker")
```

```
smoker_types = smoker_types.astype(int)

region_types = pd.get_dummies(data1.region, prefix = "region")

region_types = region_types.astype(int)
```

* Combine with the original dataset.

```
data = pd.concat([data1, sex_types, smoker_types, region_types], axis = 1)
```

* See the first rows of the dataset.

```
data.head()
```

* Drop sex_female and smoker_no because when if the female is 1, it means male is 0, same for smoker.

* All values became numerical.

```
data.drop(['sex', 'smoker', 'region', 'sex_female', 'smoker_no'], axis=1, inplace=True)

print(data)
```

* x = independent, age, bmi, children, sex, smoker, region.

* y = dependent, charges.

* We will make a prediction about y depend on x.

```
y = data['charges']

data.drop(['charges'], axis=1, inplace=True)

x = data
```

* Print independent values (x).

```
print(x.to_string)
```

* Print dependent values (y).

```
print(y.to_string())
```

* Print all dataset first rows.

```
data.head()
```

* Print unique values for bmi.

```
sorted(data['bmi'].unique())
```

* Divide values train and test.

* %25 of values will be for test, 75% will be for train.

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=46)
```

* For example age is 18, charges is 16754. we need to put both values should be between 0 and 1.

* It calls normalization.

```
scaler = StandardScaler()

x_train = scaler.fit_transform(x_train)

x_test = scaler.fit_transform(x_test)
```

* MACHINE LEARNING AND MODEL PERFORMANCE

* TREE REGRESSION MODEL

```
tree_regression = DecisionTreeRegressor(random_state=42, max_depth=2)

tree_regression = tree_regression.fit(x_train, y_train)

predict_tree_regression = tree_regression.predict(x_test)
```

* RANDOM FOREST MODEL

```
random_regression = RandomForestRegressor(n_estimators=100, max_depth=4,
random_state=42)

random_regression.fit(x_train, y_train)

predict_random_regression = random_regression.predict(x_test)
```

* LASSO REGRESSION

```
lassoReg = Lasso(alpha=2)

lassoReg.fit(x_train, y_train)

predict_lasso = lassoReg.predict(x_test)
```

* ELASTIC REGRESSION

```
elastic_reg = ElasticNet(random_state=0)

elastic_reg.fit(x_train, y_train)

predict_elastic = elastic_reg.predict(x_test)
```

* RIDGE REGRESSION

```
ridge_reg = Ridge()

ridge_reg.fit(x_train, y_train)

predict_ridge = ridge_reg.predict(x_test)
```

* RESULTS OF TESTS

```
predicts = [predict_tree_regression, predict_random_regression, predict_lasso,
predict_elastic, predict_ridge]

algoritma_names = ['Predict Tree Regression', 'Predict Random Regression', 'Predict Lasso',
'Predict Elastic', 'Predict Ridge']
```

* CALCULATED FUNCTION

```
def performance_calculate(predict):

  mae = mean_absolute_error(y_test, predict)

  mse = mean_squared_error(y_test, predict)

  rmse = np.sqrt(mse) # or mse**(0.5)

  r2 = r2_score(y_test, predict)


  data = [mae, mse, rmse, r2]

  return(data)
```

* PRINT THE RESULTS

```
series = []

metrics = ['Mean Absolute Error (MAE)', 'Mean Squared Error (MSE)', 'Root Mean Squared Erro (RMSE)', 'R2']

for i in predicts:

  data = performance_calculate(i)

  series.append(data)

from IPython.display import HTML

df = pd.DataFrame(data = series, index = algoritma_names, columns=metrics)

pd.set_option('display.colheader_justify', 'center') # center column names

print(df.to_string)
```

* RESULTS

* R2 should be between 0 And 1 , and the bigger ones are better.

* Elastic regression is the worst one.

* Tree and random forest are the better one.

* You can go to regression tree calculated part and change the parameters to increase the level of succesfull.

* Save the new sample data.

```
new_data = pd.read_csv('C:\\Users\\Asus\\OneDrive\\Masaüstü\\website\\python\\PREDICTION\\healthcare_insurance_cost_prediction\\insurance_data.csv')

new_data.head()
```

* Describe the new sample data.

```
new_data.describe()
```

* Information about the new sample data.

```
new_data.info()
```

* Check the new sample data null values.

```
print(new_data.isnull().sum())
```

* Check all the new dataset.

* People by age.

```
age_stat2 = new_data['age'].value_counts()
sex_stat2 = new_data['sex'].value_counts()
bmi_stat2 = new_data['bmi'].value_counts()
children_stat2 = new_data['children'].value_counts()
smoker_stat2 = new_data['smoker'].value_counts()
region_stat2 = new_data['region'].value_counts()
print(age_stat2)
```

* People by gender.

```
print(sex_stat2)
```

* People by bmi.

```
print(bmi_stat2)
```

* People by children.

```
print(children_stat2)
```

* People by smoking.

```
print(smoker_stat2)
```

* People by region.

```
print(region_stat2)
```

* Dataset looks it has not any problem.

* Transfrom cathegorical values to nominal values.

* Transform male and female to 0 and 1. same for smoker and region too.

```
sex_types2 = pd.get_dummies(new_data.sex, prefix = "sex")

sex_types2 = sex_types2.astype(int)

smoker_types2 = pd.get_dummies(new_data.smoker, prefix = "smoker")

smoker_types2 = smoker_types2.astype(int)

region_types2 = pd.get_dummies(new_data.region, prefix = "region")

region_types2 = region_types2.astype(int)
```

* Combine with the original dataset.

```
new_data = pd.concat([new_data, sex_types2, smoker_types2, region_types2], axis = 1)

new_data.head()
```

* Drop sex_female and smoker_no because when if the female is 1, it means male is 0, same for smoker.

* All values should be numerical.

```
new_data.drop(['sex', 'smoker', 'region', 'sex_female', 'smoker_no'], axis=1, inplace=True)

print(new_data)
```

* Drop the last column (adjust based on your extra feature location).

* Now you can make predictions.

```
X_test = new_data

X_test = X_test.iloc[:, :-1]

predicted_charges = random_regression.predict(X_test)
```

* Create a new column 'charges' in dataset2 and assign the predicted values.

* Display the modified dataset2.

```
new_data['charges'] = predicted_charges

print(new_data.head())
```

* Convert 'smoker_yes' to 'yes' and 'no'.

* Drop the original 'sex_male' and 'smoker_yes' columns.

* Display the modified dataset.

```
new_data['sex'] = new_data['sex_male'].replace({1: 'male', 0: 'female'})

new_data['smoker'] = new_data['smoker_yes'].replace({1: 'yes', 0: 'no'})

new_data = new_data.drop(columns=['sex_male', 'smoker_yes'])

print(new_data.head())
```

* Convert the 'region' column to object type.

* Optionally, drop the original region columns if no longer needed.

* Display the modified dataset.

```
new_data['region'] = new_data[['region_northeast', 'region_northwest', 'region_southeast',
'region_southwest']].idxmax(axis=1)


new_data['region'] = new_data['region'].astype(str)


new_data['region'] = new_data['region'].str.replace('region_', '', regex=False)



new_data = new_data.drop(columns=['region_northeast', 'region_northwest',
'region_southeast', 'region_southwest'])


new_data = new_data[['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges']]


new_data.head()

new_data.to_csv('new_data.csv', index=False)
```