import warnings

warnings.filterwarnings('ignore')

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

insurance= pd.read\_csv("/kaggle/input/insurance/insurance.csv")

insurance.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

insurance.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

insurance.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

insurance.shape

(1338, 7)

Point to note

Dataset has 1338 rows and 7 columns.

Looking at the data, there seems to be some fields that are categorical in nature, but in integer/float type.

We will analyse and finalize whether to convert them to categorical or treat as integer.

DATA QUALITY CHECK

Check for NULL/MISSING values

# percentage of missing values in each column

round(100\*(insurance.isnull().sum()/len(insurance)),2).sort\_values(ascending = False)

charges 0.0

region 0.0

smoker 0.0

children 0.0

bmi 0.0

sex 0.0

age 0.0

dtype: float64

# percentage of missing values in each row

round(100\*(insurance.isnull().sum(axis=1)/len(insurance)),2).sort\_values(ascending = False)[:5]

1337 0.0

439 0.0

441 0.0

442 0.0

443 0.0

dtype: float64

Finding

There are no missing / Null values either in columns or rows

Duplicate Check

insurance\_dub=insurance.copy()

# Checking for duplicates and dropping the entire duplicate row if any

insurance\_dub.drop\_duplicates(subset=None, inplace=True)

insurance\_dub.shape

(1337, 7)

insurance.shape

(1338, 7)

Insights

-The shape after running the drop duplicate command is same as the original dataframe.

-Hence we can conclude that there were zero duplicate values in the dataset.

Data Cleaning

Checking value\_counts() for entire dataframe.

This will help to identify any Unknow/Junk values present in the dataset.

for col in insurance:

print(insurance[col].value\_counts(ascending=False), '\n\n\n')

18 69

19 68

51 29

45 29

46 29

47 29

48 29

50 29

52 29

20 29

26 28

54 28

53 28

25 28

24 28

49 28

23 28

22 28

21 28

27 28

28 28

31 27

29 27

30 27

41 27

43 27

44 27

40 27

42 27

57 26

34 26

33 26

32 26

56 26

55 26

59 25

58 25

39 25

38 25

35 25

36 25

37 25

63 23

60 23

61 23

62 23

64 22

Name: age, dtype: int64

male 676

female 662

Name: sex, dtype: int64

32.300 13

28.310 9

31.350 8

30.800 8

30.875 8

..

20.100 1

29.545 1

37.900 1

25.520 1

24.090 1

Name: bmi, Length: 548, dtype: int64

0 574

1 324

2 240

3 157

4 25

5 18

Name: children, dtype: int64

no 1064

yes 274

Name: smoker, dtype: int64

southeast 364

southwest 325

northwest 325

northeast 324

Name: region, dtype: int64

1639.56310 2

8825.08600 1

43254.41795 1

5257.50795 1

1980.07000 1

..

12269.68865 1

6360.99360 1

7371.77200 1

16420.49455 1

34254.05335 1

Name: charges, Length: 1337, dtype: int64

Insights

There seems to be no Junk/Unknown values in the entire dataset.

Creating Dummy Variables

We will create DUMMY variables for 4 categorical variables 'sex','children','smoker' & 'region'.

Before creating dummy variables, we will have to convert them into 'category' data types.

# Check the datatypes before convertion

insurance.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

# Convert to 'category' data type

insurance['sex']=insurance['sex'].astype('category')

insurance['smoker']=insurance['smoker'].astype('category')

insurance['children']=insurance['children'].astype('category')

insurance['region']=insurance['region'].astype('category')

# This code does 3 things:

# 1) Create Dummy variable

# 2) Drop original variable for which the dummy was created

# 3) Drop first dummy variable for each set of dummies created.

insurance\_new = pd.get\_dummies(insurance, drop\_first=True)

insurance\_new.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1338 entries, 0 to 1337

Data columns (total 13 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 age 1338 non-null int64

1 bmi 1338 non-null float64

2 charges 1338 non-null float64

3 sex\_male 1338 non-null uint8

4 children\_1 1338 non-null uint8

5 children\_2 1338 non-null uint8

6 children\_3 1338 non-null uint8

7 children\_4 1338 non-null uint8

8 children\_5 1338 non-null uint8

9 smoker\_yes 1338 non-null uint8

10 region\_northwest 1338 non-null uint8

11 region\_southeast 1338 non-null uint8

12 region\_southwest 1338 non-null uint8

dtypes: float64(2), int64(1), uint8(10)

memory usage: 44.6 KB

insurance\_new.shape

(1338, 13)

SPLITTING THE DATA

Splitting the data to Train and Test: - We will now split the data into TRAIN and TEST (70:30 ratio)

We will use train\_test\_split method from sklearn package for this

# Check the shape before spliting

insurance\_new.shape

(1338, 13)

# Check the info before spliting

insurance\_new.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1338 entries, 0 to 1337

Data columns (total 13 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 age 1338 non-null int64

1 bmi 1338 non-null float64

2 charges 1338 non-null float64

3 sex\_male 1338 non-null uint8

4 children\_1 1338 non-null uint8

5 children\_2 1338 non-null uint8

6 children\_3 1338 non-null uint8

7 children\_4 1338 non-null uint8

8 children\_5 1338 non-null uint8

9 smoker\_yes 1338 non-null uint8

10 region\_northwest 1338 non-null uint8

11 region\_southeast 1338 non-null uint8

12 region\_southwest 1338 non-null uint8

dtypes: float64(2), int64(1), uint8(10)

memory usage: 44.6 KB

from sklearn.model\_selection import train\_test\_split

# We should specify 'random\_state' so that the train and test data set always have the same rows, respectively

np.random.seed(0)

df\_train, df\_test = train\_test\_split(insurance\_new, train\_size = 0.70, test\_size = 0.30, random\_state = 100)

Verify the info and shape of the dataframes after split:

df\_train.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 936 entries, 966 to 792

Data columns (total 13 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 age 936 non-null int64

1 bmi 936 non-null float64

2 charges 936 non-null float64

3 sex\_male 936 non-null uint8

4 children\_1 936 non-null uint8

5 children\_2 936 non-null uint8

6 children\_3 936 non-null uint8

7 children\_4 936 non-null uint8

8 children\_5 936 non-null uint8

9 smoker\_yes 936 non-null uint8

10 region\_northwest 936 non-null uint8

11 region\_southeast 936 non-null uint8

12 region\_southwest 936 non-null uint8

dtypes: float64(2), int64(1), uint8(10)