Practical No. 02

Code:

```
## Type casting #################
import pandas as pd

data = pd.read_csv("C:/Data/ethnic diversity.csv")
data.dtypes

help(data.astype)
# Now we will convert 'float64' into 'int64' type.
data.Salaries = data.Salaries.astype('int64')
data.dtypes

data.age = data.age.astype('float32')
data.dtypes

### Identify duplicates records in the data ###
data = pd.read_csv("C:/Data/mtcars_dup.csv")

duplicate = data.duplicated()
duplicate
sum(duplicate)

# Removing Duplicates
data1 = data.drop_duplicates()

############### Outlier Treatment ###############
import pandas as pd
import numpy as np
import seaborn as sns

df = pd.read_csv("C:/Data/ethnic diversity.csv")
df.dtypes

# let's find outliers in Salaries
sns.boxplot(df.Salaries)

sns.boxplot(df.age)
```

```
# No outliers in age column

# Detection of outliers (find limits for salary based on IQR)
IQR = df['Salaries'].quantile(0.75) - df['Salaries'].quantile(0.25)
print(IQR)
lower_limit = df['Salaries'].quantile(0.25) - (IQR * 1.5)
upper_limit = df['Salaries'].quantile(0.75) + (IQR * 1.5)

#### ########### 1. Remove (let's trim the dataset) ################
# Trimming Technique
# let's flag the outliers in the data setc
outliers_df = np.where(df['Salaries'] > upper_limit, True, np.where(df['Salaries'] < lower_limit,
True, False))
print(outliers_df)
sum(outliers_df)
df_trimmed = df.loc[~(outliers_df), ]
df.shape, df_trimmed.shape

# let's explore outliers in the trimmed dataset
sns.boxplot(df_trimmed.Salaries)
# we see no outiers

############### 2.Replace ###############
# Now let's replace the outliers by the maximum and minimum limit
df['df_replaced'] = pd.DataFrame(np.where(df['Salaries'] > upper_limit, upper_limit,
np.where(df['Salaries'] < lower_limit, lower_limit, df['Salaries'])))
sns.boxplot(df.df_replaced)

############### 3. Winsorization ###############
# pip install feature_engine   # install the package
from feature_engine.outliers import Winsorizer
winsor = Winsorizer(capping_method='iqr', # choose  IQR rule boundaries or gaussian for mean
and std
                    tail='both', # cap left, right or both tails
                    fold=1.5,
                    variables=['Salaries'])

df_t = winsor.fit_transform(df[['Salaries']])

# we can inspect the minimum caps and maximum caps
```

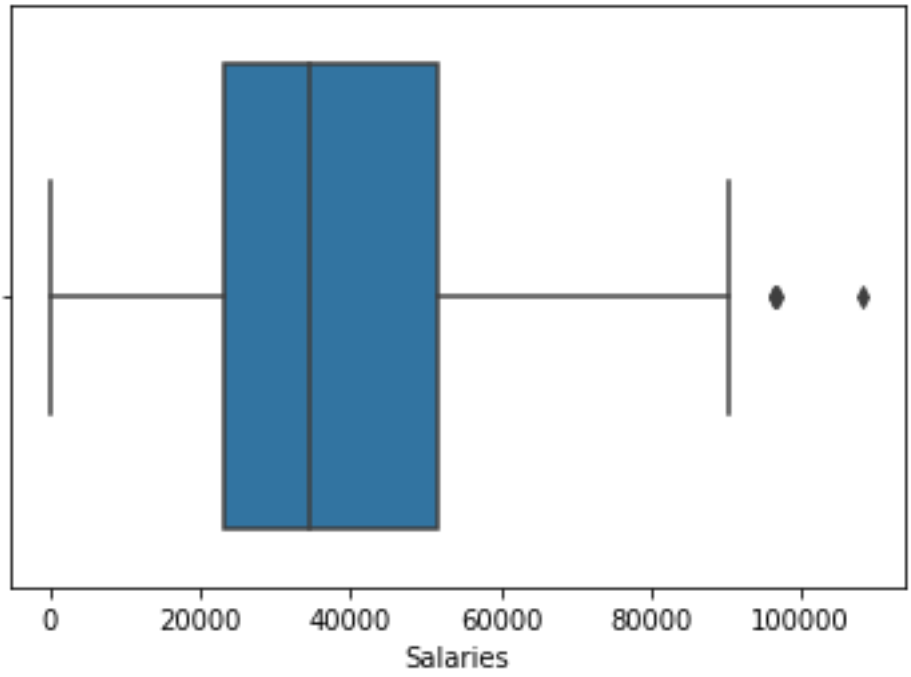
```
# winsor.left_tail_caps_, winsor.right_tail_caps_

# lets see boxplot
sns.boxplot(df_t.Salaries)

#### zero variance and near zero variance ######

# If the variance is low or close to zero, then a feature is approximately
# constant and will not improve the performance of the model.
# In that case, it should be removed.

df.var() # variance of numeric variables

##################### Missing Values Imputation ##################################
import numpy as np
import pandas as pd

# load the dataset
# use modified ethnic dataset
df = pd.read_csv('C:/Data/modified ethnic.csv') # for doing modifications

# check for count of NA'sin each column
df. sum()

# Create an imputer object that fills 'Nan' values
# Mean and Median imputer are used for numeric data (Salaries)
# Mode is used for discrete data (ex: Position, Sex, MaritalDesc)

# for Mean, Meadian, Mode imputation we can use Simple Imputer or df.fillna()
from sklearn.impute import SimpleImputer

# Mean Imputer
mean_imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
df["Salaries"] = pd.DataFrame(mean_imputer.fit_transform(df[["Salaries"]]))
df["Salaries"].isna().sum()

# Median Imputer
median_imputer = SimpleImputer(missing_values=np.nan, strategy='median')
df["age"] = pd.DataFrame(median_imputer.fit_transform(df[["age"]]))
df["age"].isna().sum()  # all 2 records replaced by median
```

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

# Mode Imputer
mode_imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
df["Sex"] = pd.DataFrame(mode_imputer.fit_transform(df[["Sex"]]))
df["MaritalDesc"] = pd.DataFrame(mode_imputer.fit_transform(df[["MaritalDesc"]]))
df.isnull().sum()  # all Sex, MaritalDesc records replaced by mode

############
# Discritization

import pandas as pd
data = pd.read_csv("C:/Data/ethnic diversity.csv")
data.head()
data.describe()
data['Salaries_new'] = pd.cut(data['Salaries'], bins=[min(data.Salaries) - 1,
                                    data.Salaries.mean(), max(data.Salaries)],
labels=["Low","High"])
data.head()
data.Salaries_new.value_counts()

################## Dummy Variables ###############
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# we use ethinc diversity dataset
df = pd.read_csv("C:/Data/ethnic diversity.csv")

df.columns # column names
df.shape # will give u shape of the dataframe

# drop emp_name column
df.drop(['Employee_Name','EmpID','Zip'], axis=1, inplace=True)
df.dtypes

# Create dummy variables
df_new = pd.get_dummies(df)
df_new_1 = pd.get_dummies(df, drop_first = True)
```

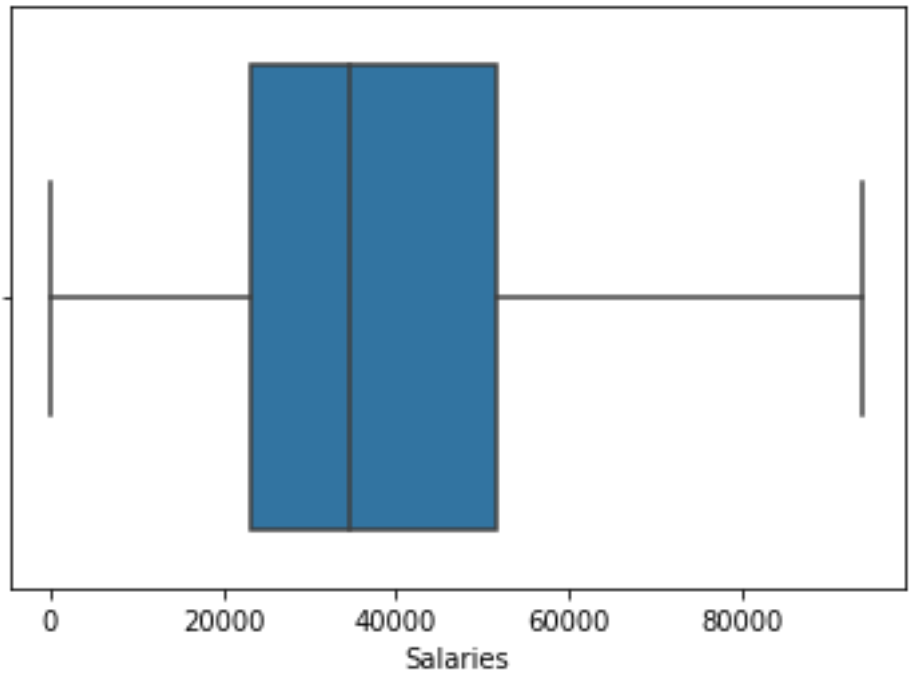
```
# we have created dummies for all categorical columns

##### One Hot Encoding works
df.columns
df = df[['Salaries', 'age', 'Position', 'State','Sex',
       'MaritalDesc', 'CitizenDesc', 'EmploymentStatus', 'Department','Race']]


from sklearn.preprocessing import OneHotEncoder
# Creating instance of One Hot Encoder
enc = OneHotEncoder() # initializing method

enc_df = pd.DataFrame(enc.fit_transform(df.iloc[:, 2:]).toarray())

# Label Encoder
from sklearn.preprocessing import LabelEncoder
# creating instance of labelencoder
labelencoder = LabelEncoder()

# Data Split into Input and Output variables
X = df.iloc[:, 0:9]

y = df['Race']
y = df.iloc[:, 9:] # Alternative approach

df.columns

X['Sex']= labelencoder.fit_transform(X['Sex'])
X['MaritalDesc'] = labelencoder.fit_transform(X['MaritalDesc'])
X['CitizenDesc'] = labelencoder.fit_transform(X['CitizenDesc'])

### label encode y ###
y = labelencoder.fit_transform(y)
y = pd.DataFrame(y)

### we have to convert y to data frame so that we can use concatenate function
# concatenate X and y
df_new = pd.concat([X, y], axis =1)

## rename column name
```

```
df_new.columns
df_new = df_new.rename(columns={0:'Race'})

import pandas as pd
import numpy as np

### Standardization
from sklearn.preprocessing import StandardScaler
d = pd.read_csv("C:/Data/mtcars.csv")

a = d.describe()
# Initialise the Scaler
scaler = StandardScaler()
# To scale data
df = scaler.fit_transform(d)
# Convert the array back to a dataframe
dataset = pd.DataFrame(df)
res = dataset.describe()

### Normalization
## load data set
ethnic = pd.read_csv("C:/Data/ethnic diversity.csv")
ethnic.columns
ethnic.drop(['Employee_Name', 'EmpID', 'Zip'], axis = 1, inplace = True)

a1 = ethnic.describe()

# get dummies
ethnic = pd.get_dummies(ethnic, drop_first = True)

### Normalization function - Custom Function
# Range converts to: 0 to 1
def norm_func(i):
    x = (i-i.min())/(i.max()-i.min())
    return(x)

df_norm = norm_func(ethnic)
b = df_norm.describe()
```

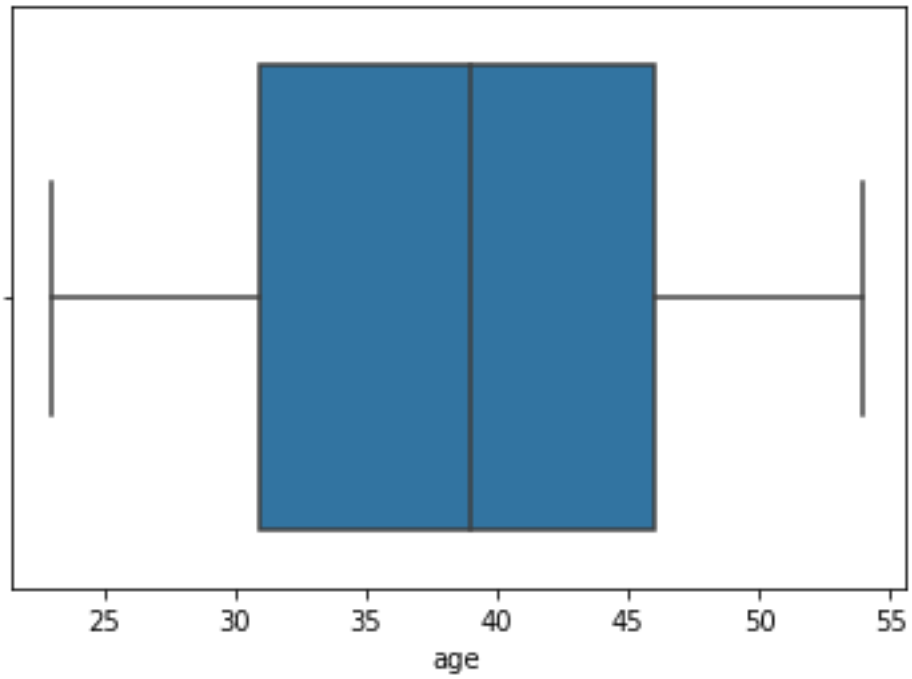
Outputs:



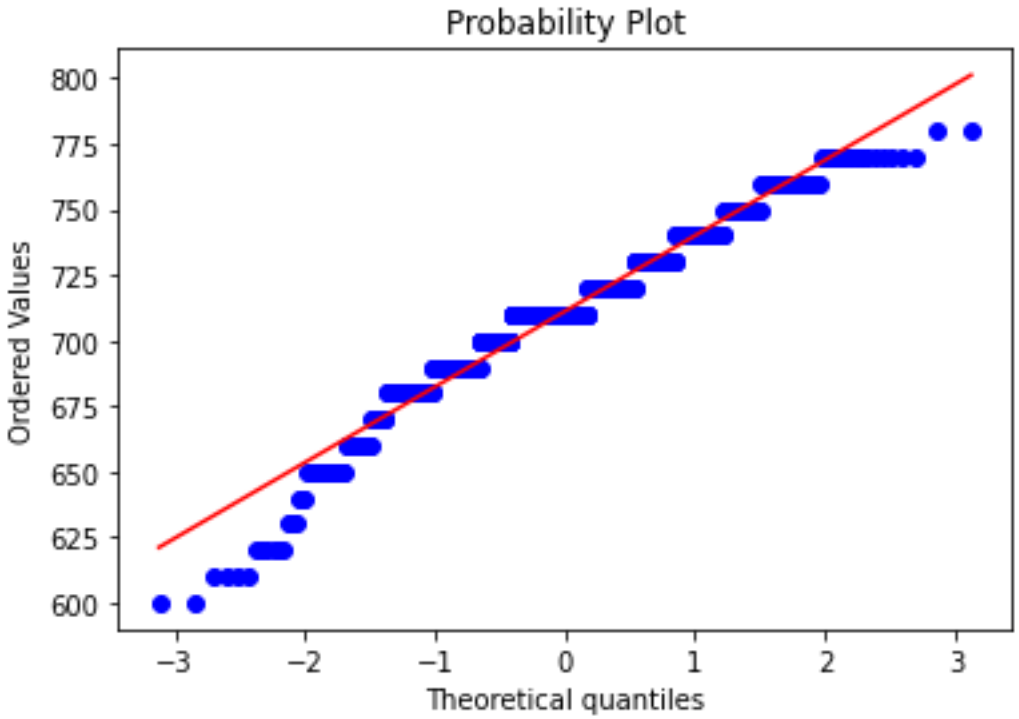
Box-Plot shows the Outliers in ethnic diversity. Salaries



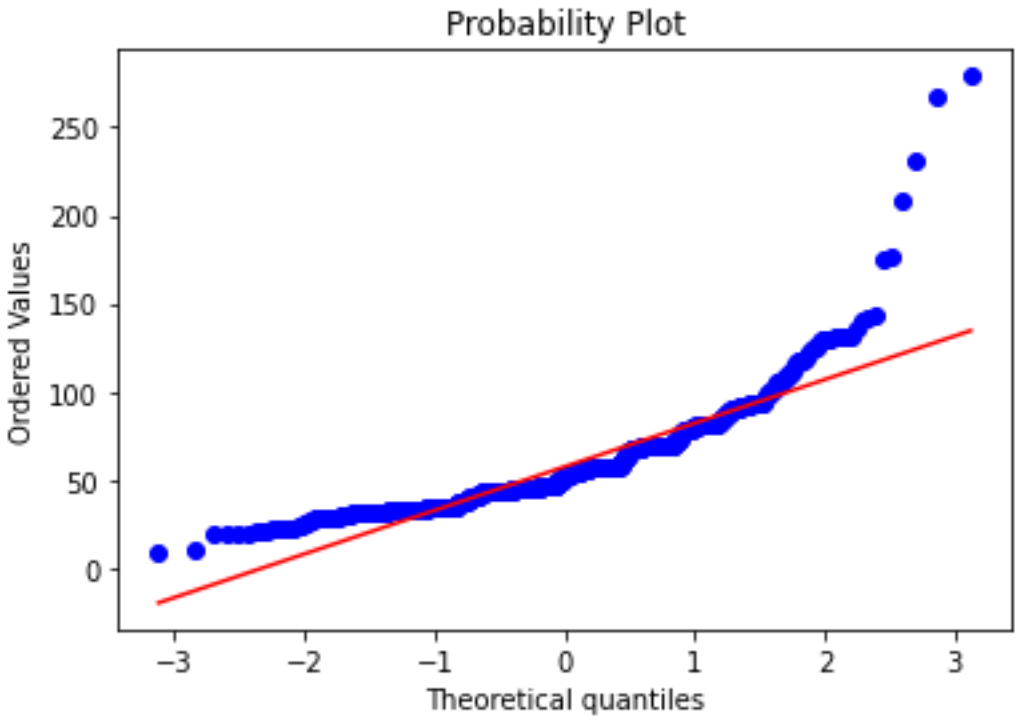
Box-Plot shows the there is no Outliers in ethnic diversity. Salaries after applying Winsorization



Box-Plot shows the Outliers in ethnic diversity. Age



Prob-plot shows the Distribution of education .gmat



Prob-plot shows the Distribution of education. workex