FUTURE SALES PREDICTION

PHASE 3

INTRODUCTION

1.Importing the required libraries(data.csv)

2.Importing the data set (read data set, create matrix)

3.Handling missing data(sklearn.preprocessing library contains class called imputer, helps in missing data)

4.Encoding catagorcial data(one_hod encoding)

5.Spliting the data set into test set and training set(import train_set_split)(X_train, X_test, Y_train, Y_test)

6.Feature scaling(import Standardscalar)

FUTURE SALES PREDICTION

1.IMPORTING THE REQUIRED LIBRARIES:

To work on future sales prediction in data science, you will typically need to use libraries in Python, as Python is a popular language for data science and machine learning. Here are some essential libraries and steps for future sales prediction:

Python: Make sure you have Python installed on your system. You can download it from the official Python website (https://www.python.org/).

NumPy: NumPy is used for numerical operations in Python. You can install it using pip:

```
Pip install numpy
```

Pandas: Pandas is used for data manipulation and analysis. Install it with pip:

```
Pip install pandas
```

Matplotlib and seaborn: These libraries are essential for data visualization:

Pip install matplotlib seaborn

Scikit_Learn: Scikit_learnis a powerful library for machine learning. You can use it for building predictive models:

Pip install scikit-learn

Tensorflow Or Torch: If you plan to work with deep learning models, you'll need one of these Libraries:

Pip install Tensorflow

# OR

Pip install torch

6.statsmodels: Statsmodels is useful for statistical modeling and

Hypothesis testing:

Pip install statsmodels

7.prophet:If you want to use a library specifically designed for

Data science and model development install it:

Pip install fbprophet

8.Jupyter notebook: While not a library, Jupyter Notebook is

An excellent interactive environment for data science and

Model development. Install it with:

Pip install jupyter

9.Others data processing libraries: Depending on your data, you might need other libraries like scikit-learn's preprocessing module (`sklearn.preprocessing`) for data preprocessing tasks.

10.Database Libraries: your data is stored in a database, consider using libraries like

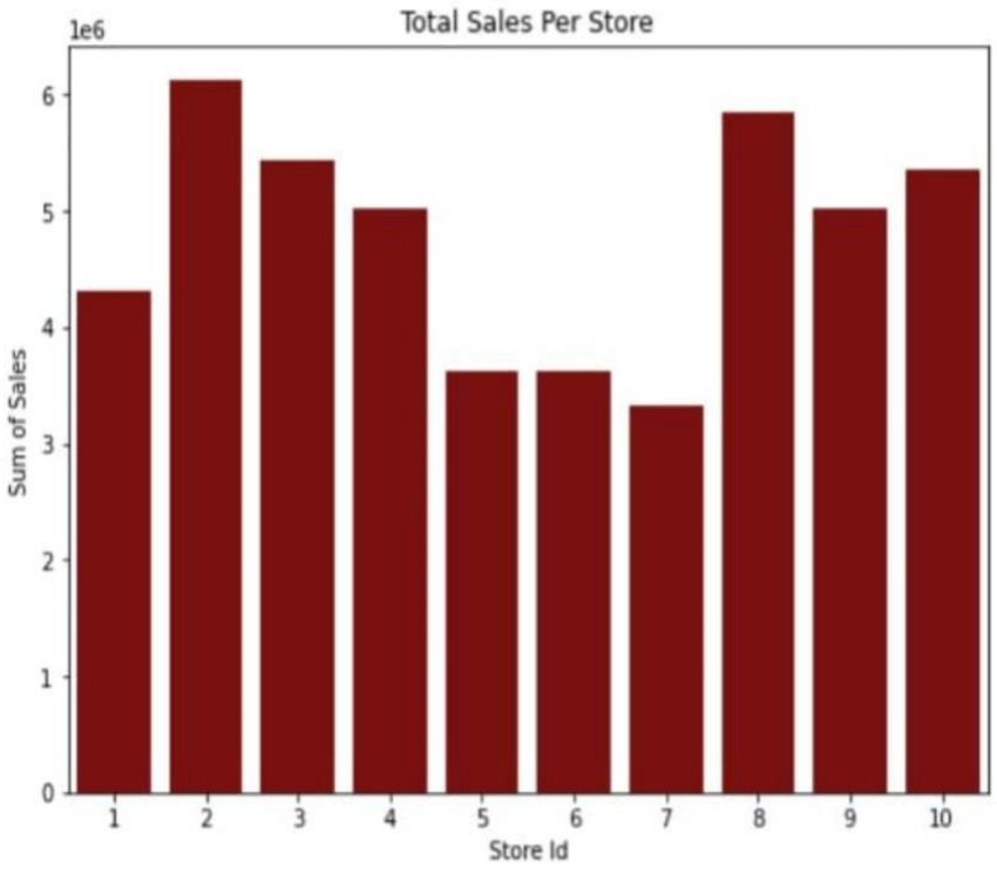
SQLAlchemy or PyMySQL to connect to and fetch data from the database.

Remember to keep your libraries and dependencies up-to-date. N do this by periodically running.

Pip Install-upgrade library-name. Once you have these libraries installed, you can start working on your future sales prediction project using the various data preprocessing, modeling, and evaluation .

2.IMPORTING THE DATA SET(READ DATA SET, CREATE MATRIX)

First step is to load the data and transform it into a structure that we will then use for each of our models. In its raw form, each row of data



Represents a single day of sales at one of ten stores. Our goal is to predict monthly sales, so we will first consolidate all stores and days into total monthly sales.

Dataset =
pd.read_csv(‘https://raw.githubusercontent.co

m/amankharwal/WebsiteData/master/advertisi
ng.csv’)

Df= dataset.copy()

Df.head()

Idsales

052

```
   1          52
   2
```

252

352

452

Def load_data(file_name):

“””Returns a pandas dataframe from a csv
file.”””””

Return pd.read_csv(file_name)

Sales_data = load_data(‘../input/demand-
forecasting-kernels-only/train.csv’)

```
Df_s.sales_data.copy(
Df_s.info()
<class ‘pandas.core.frame.DataFrame”>
RangeIndex: 913000 entries, 0 to 912999
Data columns (total 4 columns):
#   Column  Non-Null Count   Dtype
---  ------  --------------   -----
Date    913000 non-null  object
Store   913000 non-null  int64
Item    913000 non-null  int64
Sales   913000 non-null  int64
Dtypes: int64(3), object(1)
Memory usage: 27.9+ MB
Df_s.tail()
Date     store    item sales
912995 2017-12-27 1050 63
```

```
912996 2017-12-28 10.     50.     59 912997
2017-12-29 10 50 74
912998 2017-12-30 10 50 62
912999 2017-12-31 10 50 82
```

# To view basic statistical details about dataset:

```
Df_s['sales'].describe()
Count    913000.000000
Mean         52.250287
Std         28.801144
Min          0.000000
25%         30.000000
50%         47.000000
75%         70.000000
Max        231.000000
Name: sales, dtype: float64
```

Sales seem to be unbalanced!

Df_s[‘sales’].plot()

COLUMN DETAILS:

Column explanation in the dataset:

The dataset we mentioned appears to be related to predicting sales based on different advertising mediums, such as television, radio, and newspapers. Here’s a brief explanation of the columns typically found in such a dataset:

1. **Television**: This column likely contains data on the amount of money spent on advertising via television for a specific period, such as a week or month.

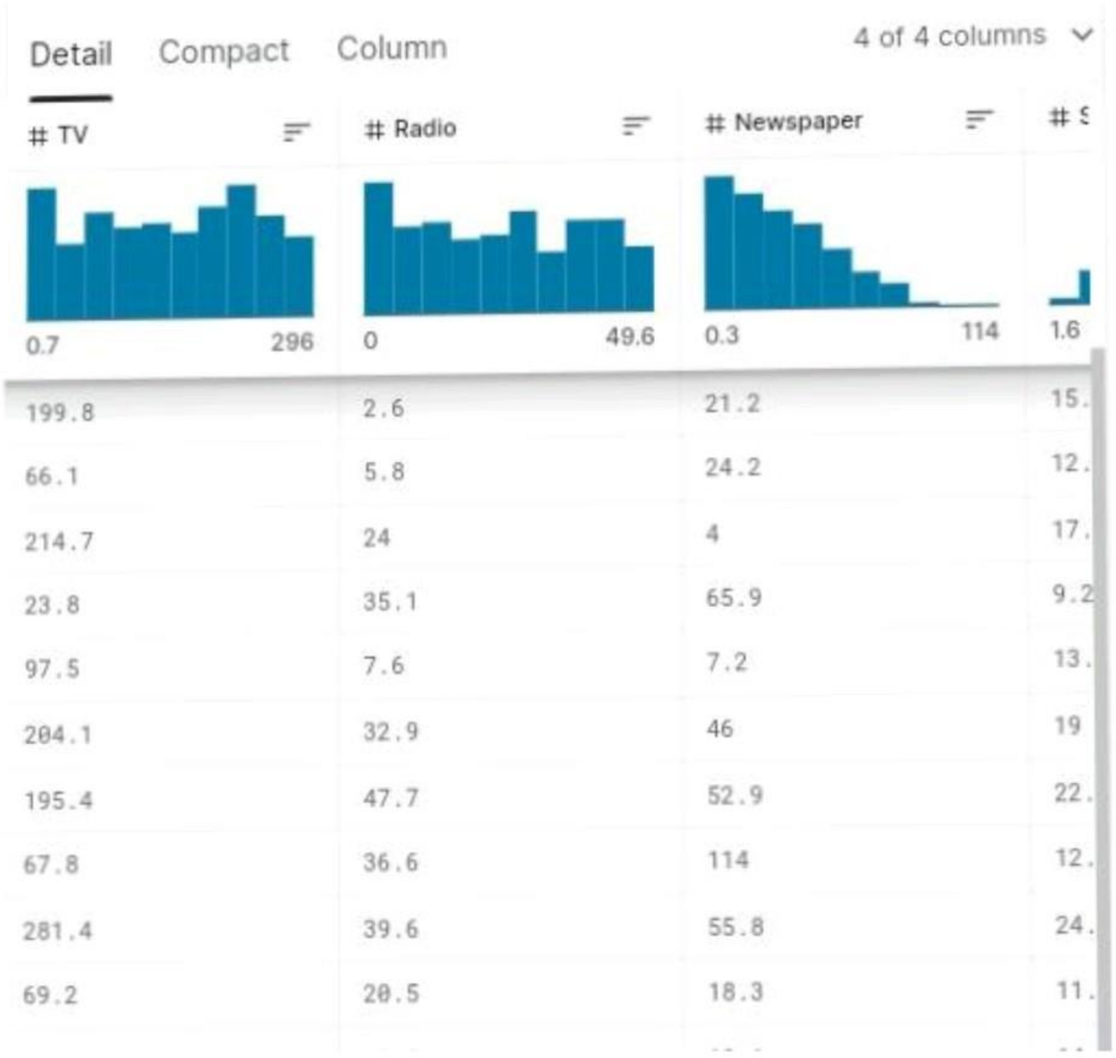
2. **Radio**: Similarly, this column contains data on advertising expenses for radio advertising.

3. **Newspaper**: This column contains data on advertising expenses for newspaper advertisements.

4. **Sales**: This column usually represents the sales figures for a product or service during the same period for which advertising expenses are recorded.

To predict future sales based on these advertising mediums, you can use regression analysis or machine learning techniques.

By analyzing historical data , we can build a predictive model that takes advertising expenses as input features (Television, Radio, Newspaper) and predicts future sales, helping businesses make informed decisions about their advertising budgets.



# TV	# Radio	# Newspaper	# S
0.7 – 296	0 – 49.6	0.3 – 114	1.6
199.8	2.6	21.2	15.
66.1	5.8	24.2	12.
214.7	24	4	17.
23.8	35.1	65.9	9.2
97.5	7.6	7.2	13.
204.1	32.9	46	19
195.4	47.7	52.9	22.
67.8	36.6	114	12.
281.4	39.6	55.8	24.
69.2	20.5	18.3	11.

3.HANDLING THE MISSING DATA(SKLEARN.PREPROCESSING LIBRARY CONTAINS CLASS IMPUTER, HELPS IN MISSING DATA)

1.Drop columns with missing values:

Let’s try to drop some of the columns which won’t contribute much to our machine learning model. We’ll start with Tv, Radio, Newspaper and sales.

```
Cols = [‘TV',’Radio’,’Newspaper',’Sales']
df = df.drop(cols, axis=1)
>>>df.info( )
Date    913000 non-null  object
Store   913000 non-null  int64
Item    913000 non-null  int64
Sales   913000 non-null  int64
```

2.Drop rows with missing values:

Next we can drop all rows in the data that have missing values (NaNs). Here’s how:

```
>>>df=df.dropna( )
>>>df.info( )
```

```
Int64Index: 712 entries, 0 to 890
Data columns (total 4 columns):
Date    712  non-null  object
Store   712  non-null  int64
Item    712  non-null  int64
Sales   712  non-null  int64
```

3.Take care of missing data:

Everything's clean now, except **TV**, which has lots of missing values. Let's compute a median or **interpolate()** all the ages and fill those missing age values. Pandas has an **interpolate()** function that will replace all the missing NaNs to interpolated values.

df['TV'] = df['TV'].interpolate()

Now let's observe the data columns. Notice **Tv** now interpolated with imputed new values.

>>>df.info()

```
Data columns (total 4 columns):
Date    913000 non-null  object
Store   913000 non-null  int64
Item    913000 non-null  int64
Sales   913000 non-null  int64
```

4.ENCODING CATEGORICAL DATA(ONE_HOT ENCODING)

One-hot encoding is a common technique for encoding categorical data. It involves converting categorical variables into a form that could be provided to ML algorithms to do a better job in prediction. This method creates binary columns for each category, where the presence of a category is indicated by 1 and the absence by 0.

Sure, let’s take an example to demonstrate one-hot encoding.

Suppose you have a dataset with a categorical variable “products” having three categories: “TV,” “Newspaper,” and “Radio.” One-hot encoding will create three binary columns, one for each category.

Original data:

| products |

| TV |

| Newspaper|

| Radio |

| TV |

After one-hot encoding, the data would look like this:

| Is_TV | Is_Newspaper| Is_Radio|

|--------|---------|----------|

| 1 | 0 | 0 |

0	1	0
0	0	1
1	0	0

In this encoding, each row indicates whether a specific products is sale (1) or not (0). This format is often used in machine learning algorithms to handle categorical data.

5.SPLITING THE DATA SET INTO TEST SET AND TRAINING SET(import train_test_split)(X_train, X_test, Y_train, Y_test)

Now that we're ready with X and y, let's split the data set: we'll allocate 70 percent for training and 30 percent for tests using scikit **model_selection.**

From sklearn.model_selection

import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.3, random state=0)

And that’s all, folks. Now you can preprocess data on your own. Go on and try it for yourself to start building your own models and making predictions.

6.FUTURE SCALING(Standardscalar)

When preparing data for future sales prediction in data science, using standard scaling is a common practice to ensure that all features are on the same scale. This prevents any particular feature from dominating the model due to its larger magnitude. Standard scaling, also known as Z-score normalization, transforms the data to have a mean of 0 and a standard deviation of 1. This enables more effective training of machine learning models, especially those sensitive to the scale of the features.

Example:

Sure, here’s an example of how you can use standard scaling for future sales prediction in data science:

```python
Import pandas as pd
From sklearn.preprocessing import
Standardscalar
From sklearn.model_selection import
train_test_split
From sklearn.linear_model import
LinearRegression
# Sample data
Data = {‘sales’: [100, 200, 300, 400, 500],
        ‘advertising_cost’: [50, 100, 150, 200,
250],
        ‘product_price’: [5, 6, 7, 8, 9]}
```

```
df = pd.DataFrame(data)
# Separate the target variable
X = df[[‘advertising_cost’, ‘product_price’]]
Y = df[‘sales’]
# Standard Scaling
Scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Splitting the data
X_train, X_test, y_train, y_test =
train_test_split(X_scaled, y, test_size=0.2,
random_state=42)
# Training the model
Model = LinearRegression()
Model.fit(X_train, y_train)
# Predicting on new data
New_data = {‘advertising_cost’: [120, 180],
        ‘product_price’: [6.5, 7.5]}
```

```
New_df = pd.DataFrame(new_data)
New_X_scaled = scaler.transform(new_df)
Predictions = model.predict(new_X_scaled)
Print(predictions)
```

In this example, we first create a sample dataset with sales, advertising costs, and product prices. We then use StandardScaler from the sklearn library to scale the data. After that, we split the data into training and testing sets, train a Linear Regression model, and make predictions on new data after scaling it appropriately.

CONCLUSION

This all are belongs into the future sales prediction. Using this topics ,we can easily understand the project.