APPLIED DATA SCIENCE GROUP-2

FUTURE SALES PREDICTION (PHASE - 4)

PROJECT DEVELOPMENT – 2

In this part we will continue building the project :

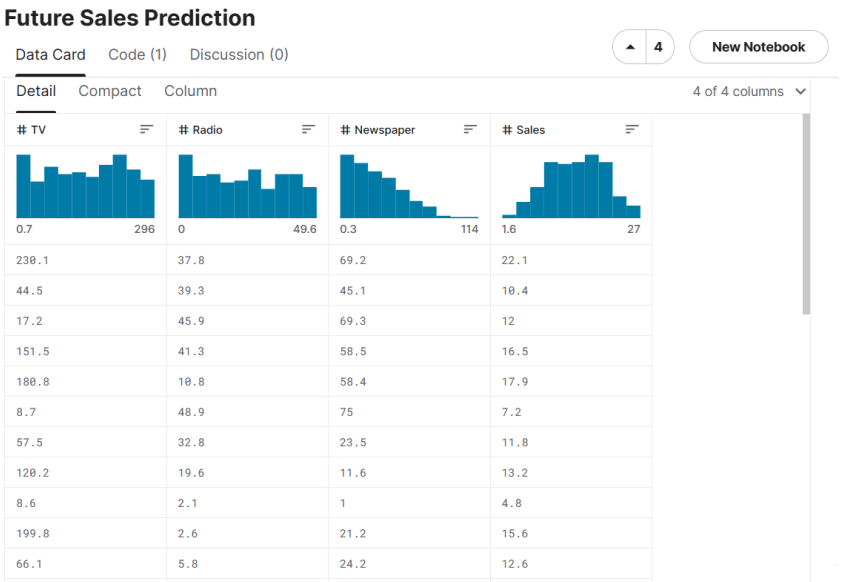
- ✓ Model training
- ✓ Feature engineering
- ✓ Evaluation

DATASET:

Dataset: www.kaggle.com

Dataset name: future-sales-prediction

Dataset link: https://www.kaggle.com/datasets/chakradharmattapalli/future-sales-prediction



MODEL TRAINING

Model training is the primary step in machine learning, resulting in a working model that can then be validated, tested, and deployed. The model's performance during training will eventually determine how well it will work when it is eventually put into an application for the end-users.

Step 1: Data preprocessing

Load the dataset and check for any missing values

```
data = pd.read_csv("F:\AIML DATASET\Sales.csv")
print(data.isnull().sum())
```

```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

Step 2: Data splitting

The first step in model training is to prepare our data. Split the dataset into training and test sets using the train_test_split function from the scikit-learn library. This ensures that we have separate data for training and evaluating our model.

```
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)
```

Step 3: Train the Model

Next, select an appropriate model for the task and train it using the training set. Let us train a logistic regression model using scikit-learn:

```
from sklearn.linear_model import LinearRegression
# Create a instance Linear Regression model
model = LinearRegression()
# Fit the model on the scaled training data
model.fit(xtrain_scaled, ytrain)
```

Step 4: Evaluate on the Test Set

Now, it's time to evaluate our model on the test set. Use the trained model to make predictions on the test data and compare them to the actual labels.

```
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
# Make predictions on the scaled test data
ypred_scaled = model.predict(xtest_scaled)
# Evaluate the model's performance on scaled data
mse_scaled = mean_squared_error(ytest, ypred_scaled)
r2_scaled = r2_score(ytest, ypred_scaled)
```

Step 5: Perform Regression metrics

Metrics refer to quantitative measures used to assess the performance, quality, or characteristics of a model. Metrics are

essential for making data-driven decisions, comparing different models and determining whether the data science project is successful.

Mean Squared Error (MSE): The average of the squared differences between predicted and actual values.

```
mse_scaled = mean_squared_error(ytest, ypred_scaled)
```

Root Mean Squared Error (RMSE): The square root of the MSE, which provides a measure of the error in the original units.

```
r2_scaled = r2_score(ytest, ypred_scaled)
```

Step 6: Assess Model's Performance

Analyze the evaluation metrics obtained from the previous steps to assess the model's performance. Consider the context of the problem and compare the results against the desired performance level or any baseline models. This analysis will provide insights into the strengths and weaknesses of the model.

Step 7: Iterate and Improve (if needed)

Based on the assessment, we may need to iterate and improve the model. Consider collecting more data, refining features, trying different algorithms, or tuning hyperparameters. Repeat the evaluation process until we achieve the desired performance.

FEATURE ENGINEERING:

Feature engineering is a process of creating new features or transforming existing features to improve the performance of a machine learning model.

There are many feature engineering technique but we use “scaling” in our project.

Scaling is a technique used to transform numerical variables to have a similar scale, so that they can be compared more easily. The most common scaling techniques are standardization and normalization. Standardization scales the variable so that it has zero mean and unit variance.

Standardization:

We use the `StandardScaler` from scikit-learn. Standardization scales the features to have a mean of 0 and a standard deviation of 1, which can improve the performance of some machine learning algorithms, especially linear models. We scale our features before fitting them to the Linear Regression model.

```
from sklearn.preprocessing import StandardScaler
# Create a StandardScaler instance
scaler = StandardScaler()
# Fit the scaler
xtrain_scaled = scaler.fit_transform(xtrain)
# Transform the test data using the same scaler
xtest_scaled = scaler.transform(xtest)
# Create a Linear Regression model
model = LinearRegression()
# Fit the model on training data
model.fit(xtrain_scaled, ytrain)
# Make predictions on test data
ypred_scaled = model.predict(xtest_scaled)
```

EVALUATION:

Evaluation in data science refers to the process of assessing the performance, accuracy, and effectiveness of a predictive model, algorithm, or analytical approach applied to a dataset. It is a critical step in the data science workflow that helps determine how well the model or analysis meets the goals and objectives of a given project.

MSE AND R-Squared value:

Mean Squared Error (MSE): The average of the squared differences between predicted and actual values.

Root Mean Squared Error (RMSE): The square root of the MSE, which provides a measure of the error in the original units.

```
# Evaluate the model's performance on scaled data
mse_scaled = mean_squared_error(ytest, ypred_scaled)
r2_scaled = r2_score(ytest, ypred_scaled)

print("Mean Squared Error (Scaled):", mse_scaled)
print("R-squared Score (Scaled):", r2_scaled)
```

```
Mean Squared Error (Scaled): 2.9077569102710923
R-squared Score (Scaled): 0.9059011844150826
```

DATA VISUALIZATION

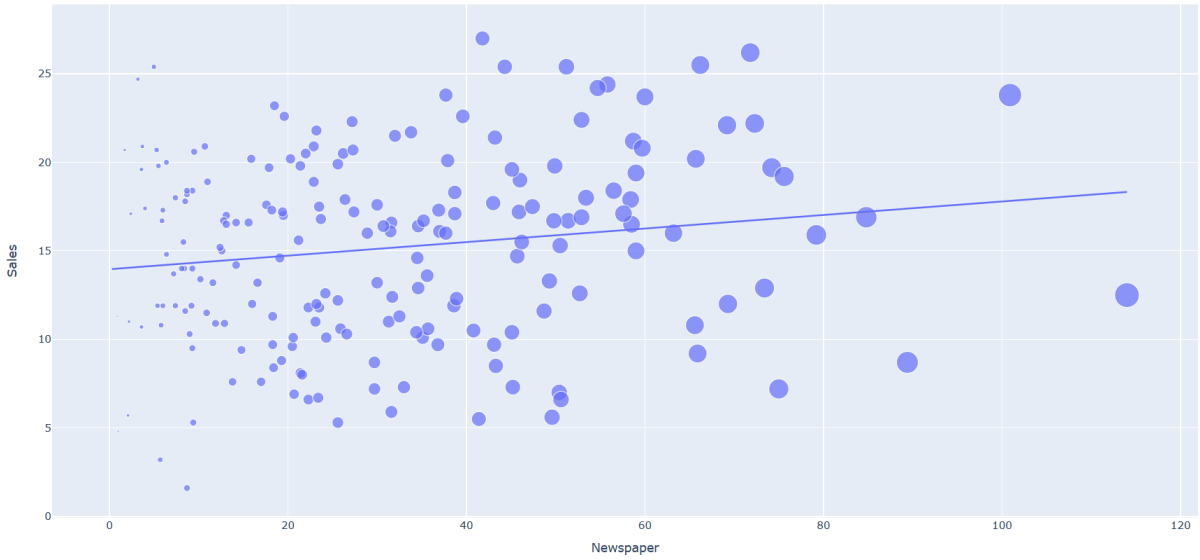
Data visualization is the representation of data through use of common graphics, such as charts, plots, infographics, and even animations. These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand.

```
import plotly.express as px
```

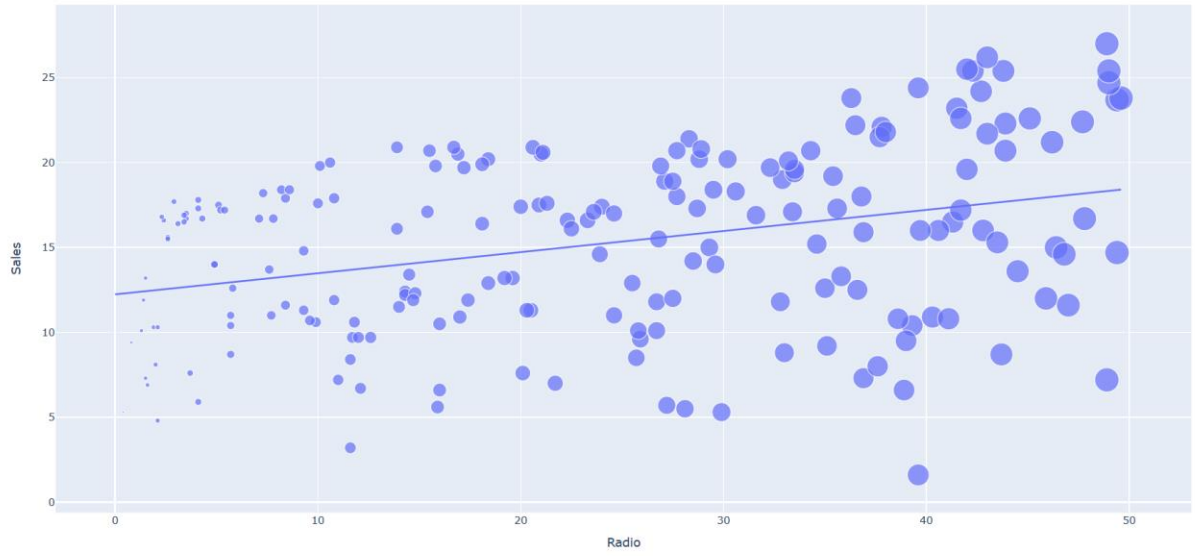
Plotly is a popular Python library for creating interactive, web-based data visualizations. Plotly Express is a high-level interface within the Plotly library that simplifies the process of creating a wide range of interactive charts, such as scatter plots, bar charts, line charts,etc. It is especially useful for rapid prototyping and exploratory data analysis.

```
# Plot representation of TV
figure = px.scatter(data_frame=data, x="TV", y="Sales", size="TV", trendline="ols")
figure.show()
```

```
# Plot representation of Newspaper
figure = px.scatter(data_frame=data, x="Newspaper", y="Sales", size="Newspaper", trendline="ols")
figure.show()
```




```
# Plot representation of Radio
figure = px.scatter(data_frame=data, x="Radio", y="Sales", size="Radio", trendline="ols")
figure.show()
```



CONCLUSION:

So this is how we can train a machine learning model to predict the future sales of a product. Predicting the future sales of a product helps a business manage the manufacturing and advertising cost of the product. Sales forecasting is done by analyzing customer purchasing behaviour and it plays an important role in modern business intelligence. Forecasting future sales demand is key to business and business planning activities. Forecasting helps business organizations to make improvements, to make changes to business plans and to provide a stock storage solution. Forecast is determined by the use of data or information from past works and the consideration of recognized feature in future. Sales forecasting plays a vital role in strategic planning and market strategy for every company to assess past and present sales statistics and predict potential results.