

APPLIED DATA SCIENCE – Phase 4

Topic:Future Sales Prediction

Team members:

K.Jayapriya - (511321104036) - [jayapriyakumaresan02@gmail.com](mailto:jayapriyakumaresan02@gmail.com)

S.Anu - (511321104005) [- s.anupriyanka1225@gmail.com](mailto:-%20%20s.anupriyanka1225@gmail.com)

M.Charishma - (511321104014) [- Charishmamasarapu@gmail.com](mailto:-%20%20Charishmamasarapu@gmail.com)

P.Lahari - (511321104045) [- laharipasala8@gmail.com](mailto:-%20%20%20laharipasala8@gmail.com)

Content:

* Sales prediction
* Feature Engineering
* Model Training
* Model Evaluation
* Data Analysis

Required packages and installation:

1.Numpy

2.Pandas

3.Keras

4.Tensor flow

5.CSV

6.Matplotlib.pyplot

Sales Forecasting:

Sales forecasting in data science involves using advanced techniques and algorithms to predict future sales based on historical data and other relevant variables. Here's how data science methods can be applied to sales forcasting

**Data Collection and Preparation:**

**Data Collection:**

**Gather historical sales data, including information on sales volume, prices, seasonality, promotions, and external factors (e.g., economic indicators, weather).**

**Data Cleaning:**

**Clean the data to handle missing values, outliers, and inconsistencies.**

**FEATURE ENGINEERING:**

**Create relevant features from the existing data, such as calculating moving averages, aggregating sales by day/week/month, and incorporating lag variables.**

**1.Imputation:**

**Handling missing data by filling in missing values using techniques like mean, median, or more advanced methods such as k-nearest neighbors imputation.**

**2. Handling Categorical Variables:**

**Converting categorical variables into numerical representations through techniques like one-hot encoding, label encoding, or target encoding.**

**Creating binary features indicating the presence or absence of specific categories (binary encoding).**

**3. Transformations:**

**Applying mathematical transformations to numerical features, such as logarithm, square root, or Box-Cox transformations, to make the data more normally distributed.**

**Binning numerical features to convert them into categorical variables, which can capture non-linear relationships.**

**4. Feature Interactions:**

**Creating new features by combining existing features. For example, if you have height and weight, a new feature could be BMI (Body Mass Index) calculated as weight divided by height squared.**

**Adding, subtracting, multiplying, or dividing relevant features to create interaction terms that might contain useful information for the model.**

**5. Temporal Features:**

**Extracting features from timestamps, such as hour of the day, day of the week, month, or year, which can capture patterns related to time.**

**Creating time-based aggregations, such as average sales per day of the week or month, can also be useful.**

**6. Text Data:**

**Extracting features from text, such as word frequency, length of the text, presence of specific keywords, or using techniques like TF-IDF (Term Frequency-Inverse Document Frequency) for text vectorization.**

**7. Domain-Specific Features:**

**Incorporating domain knowledge to create features that are specifically relevant to the problem at hand. For example, in an e-commerce context, features like discounts as a percentage of the original price can be valuable.**

**8. Feature Scaling:**

**Scaling numerical features to a similar range (e.g., using Min-Max scaling or standardization) to prevent features with larger scales from dominating the learning process.**

**9. Dimensionality Reduction:**

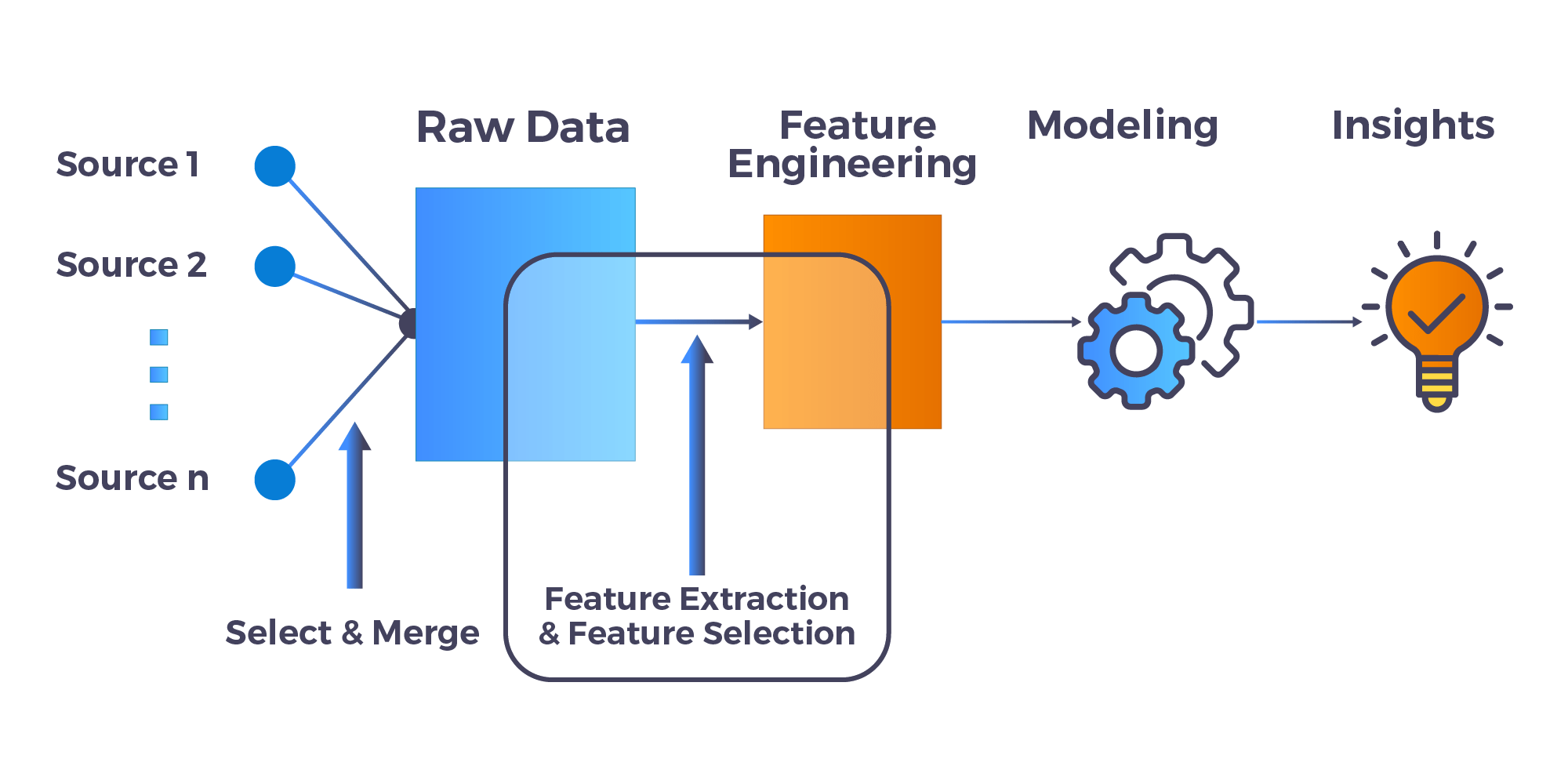
**Applying techniques like Principal Component Analysis (PCA) or t-SNE to reduce the dimensionality of the data while retaining important information.**

**10. Target Encoding:**

**Encoding categorical variables based on the mean or median of the target variable for each category. This can capture the relationship between the categorical feature and the target.**

**11. Feature Selection:**

**Using techniques like univariate feature selection, recursive feature elimination, or feature importance from tree-based models to select the most relevant features for the model**



**MODEL TRAINING:**

Training a model to predict future sales involves using historical sales data to build a predictive model that can forecast future sales.

1. Problem Definition:

Clearly define the problem:

2. Data Splitting:

Split the dataset into training and testing sets. The training set is used to train the model, and the testing set is used to evaluate its performance.

3. Model Selection:

Choose appropriate algorithms for your problem. 6. Training the Model:

Tune hyperparameters to improve the model's performance. This might involve techniques like grid search or random search.

4. Model Evaluation:

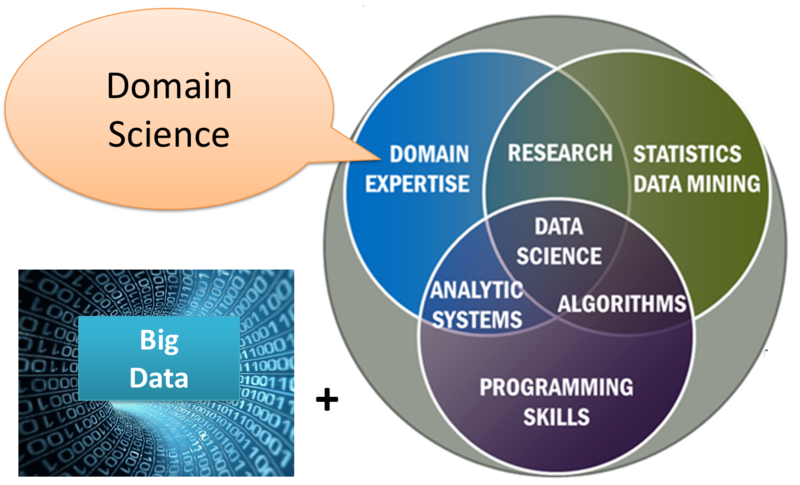
Evaluate the model's performance on the test dataset, using appropriate metrics (such as Mean Absolute Error, Root Mean Square Error, or others depending on your specific requirements).

5. Model Deployment:

Once you have a satisfactory model, deploy it to make predictions on new, unseen data.

6. Monitoring and Maintenance:

Regularly monitor the model’s performance in a real-world scenario.



**MODEL EVALUATION:**

When it comes to evaluating models in data science related to predicting future sales, there are several key metrics and techniques you can use to assess the performance of your model.

Mean Absolute Error (MAE):

Measures the average absolute errors between the actual and predicted values. It gives you an idea of the magnitude of errors.

Mean Squared Error (MSE):

Measures the average of the squares of errors. It penalizes larger errors more significantly than MAE.

Root Mean Squared Error (RMSE):

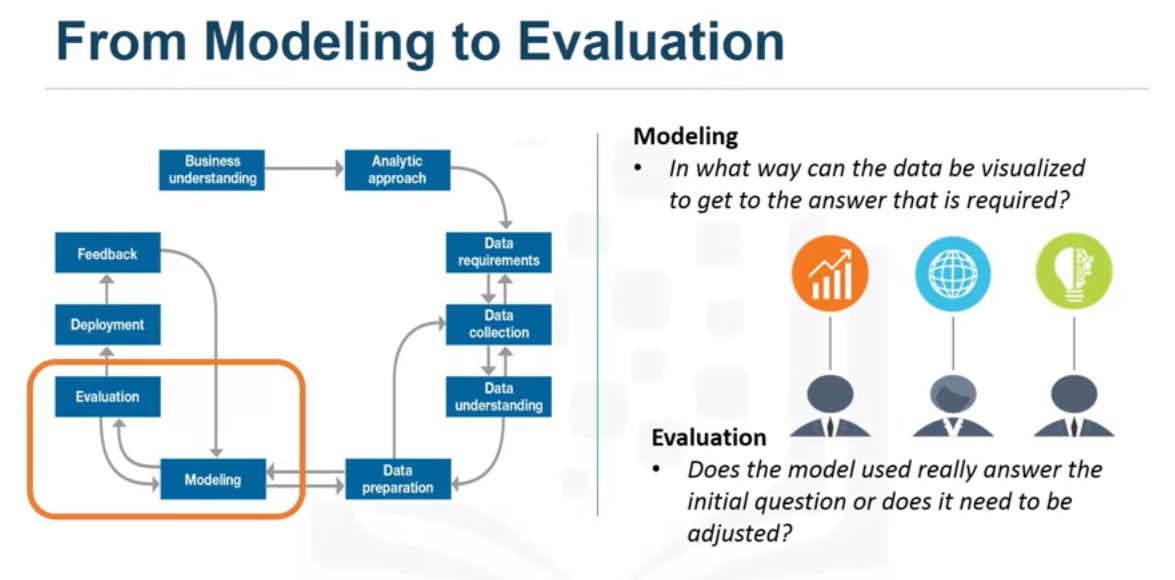
The square root of MSE, providing an interpretable scale similar to the original target variable.

R-squared (R²) Score:

Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. R² values range from 0 to 1; higher values are better.

Adjusted R-squared:

It adjusts R² for the number of predictors in the model. It's especially useful when you have multiple predictor variables.



**DATA ANALYSIS:**

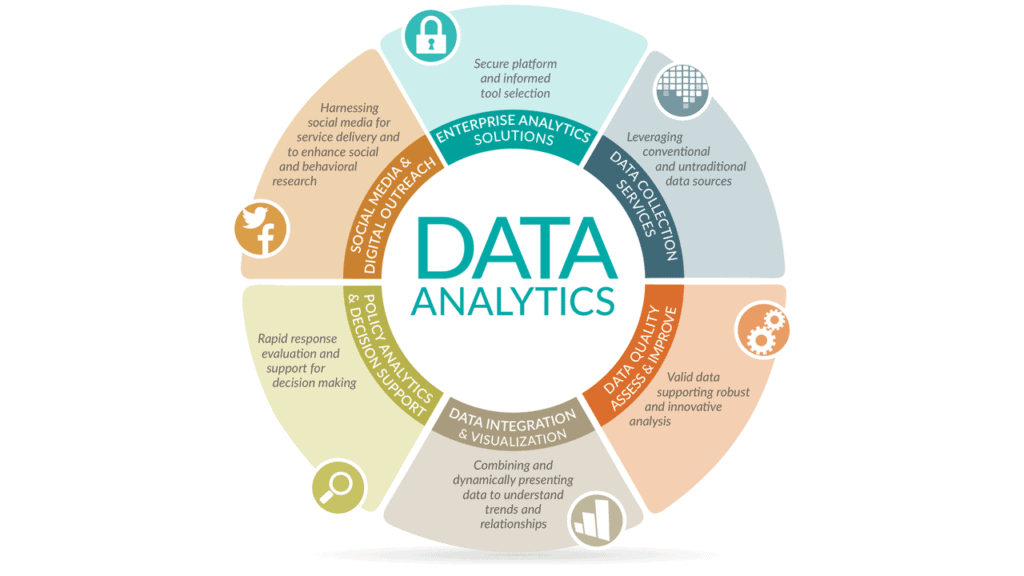
Data analysis is a process of inspecting, cleaning, transforming, and modeling data to discover useful information, draw conclusions, and support decision-making. It plays a crucial role in various fields, including business, science, social sciences, and research

Use software/tools:

R, Python (with libraries like Pandas, NumPy, SciPy), Excel, or specialized tools like SPSS or SAS.

DATA PRIVACY:

Ensure that you are handling data responsibly and ethically, especially if dealing with sensitive information.



SALES PREDICTION:

Sales prediction in data science is a crucial application that helps businesses forecast future sales based on historical data and various influencing factors. Predicting sales accurately can optimize inventory management, improve budgeting, and enhance overall business strategies.

Source code

from warnings import filterwarnings

filterwarnings('ignore')

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

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

from sklearn.linear\_model import LinearRegression, Lasso, Ridge, ElasticNet, RidgeCV, LassoCV, ElasticNetCV

from sklearn.metrics import mean\_squared\_error, r2\_score

df=pd.read\_csv("/kaggle/input/tvradionewspaperadvertising/Advertising.csv")

data=df.copy()

df.head()

df.info()

df.describe().Tdf.isnull().sum()

dtype: int64

Exploratory Data Analysis

Correlation

df.corr()

sns.heatmap(df.corr(), annot = True)

plt.title("Correlation Matrix")

plt.show()

sns.pairplot(df, x\_vars=['TV','Radio','Newspaper'], y\_vars='Sales', size = 4)

sns.pairplot(df, kind="reg")

sns.jointplot(x='TV', y='Sales', data=df, kind='reg')

sns.regplot(df['TV'], df['Sales'])

Feature Engineering

sns.distplot(df['Sales'])

sns.distplot(df['Sales'], fit=stats.norm)

X=df.drop('Sales', axis=1)

Y=df[['Sales']]

Model

def model(X, Y, algo, split\_share=0.33):

X\_train, X\_test, Y\_train, Y\_test=train\_test\_split(X, Y, test\_size=split\_share, random\_state=42)

m=algo.fit(X\_train, Y\_train)

train\_rmse=np.sqrt(mean\_squared\_error(Y\_train, m.predict(X\_train)))

test\_rmse=np.sqrt(mean\_squared\_error(Y\_test, m.predict(X\_test)))

return type(m).\_\_name\_\_, train\_rmse, test\_rmse

models=[LinearRegression(), Lasso(), Ridge(), ElasticNet()]

results={'model':[], 'train\_rmse':[], 'test\_rmse':[]}

for i in models:

res=model(X, Y, i)

results['model'].append(res[0])

results['train\_rmse'].append(res[1])

results['test\_rmse'].append(res[2])

results=pd.DataFrame(results)

def model\_tuning(X, Y, algo, algocv, alphas, split\_share=0.33):

X\_train, X\_test, Y\_train, Y\_test=train\_test\_split(X, Y, test\_size=split\_share, random\_state=42)

algo\_cv=algocv(alphas=alphas, cv=10, normalize=True)

algo\_cv.fit(X\_train, Y\_train)

algo\_tuned=algo(alpha=algo\_cv.alpha\_)

algo\_tuned.fit(X\_train, Y\_train)

train\_rmse=np.sqrt(mean\_squared\_error(Y\_train, algo\_tuned.predict(X\_train)))

test\_rmse=np.sqrt(mean\_squared\_error(Y\_test, algo\_tuned.predict(X\_test)))

return type(algo()).\_\_name\_\_, train\_rmse, test\_rmse

models={Ridge: RidgeCV, Lasso:LassoCV, ElasticNet:ElasticNetCV}

alphas=10\*\*np.linspace(10, -2, 100)\*0.5

results\_tuned={'model':[], 'train\_rmse':[], 'test\_rmse':[]}

for model in models.keys():

res=model\_tuning(X, Y, model, models[model], alphas)

results\_tuned['model'].append(res[0])

results\_tuned['train\_rmse'].append(res[1])

results\_tuned['test\_rmse'].append(res[2])

results\_tuned=pd.DataFrame(results\_tuned)

output: