Sales Forecasting

PHASE:4 PROJECT

FUTURE SALES PREDICTION

PROJECT SUBMISSION ON:26/10/2023

SUBMITTED BY,

1.J.SHALINI

2.K.R.SUDHARSNA

3.M.VINITHA

4.R.YAMINI

COLLEGE NAME:SACS MAVMM ENGINEERING COLLEGE

COLLEGE CODE:9123

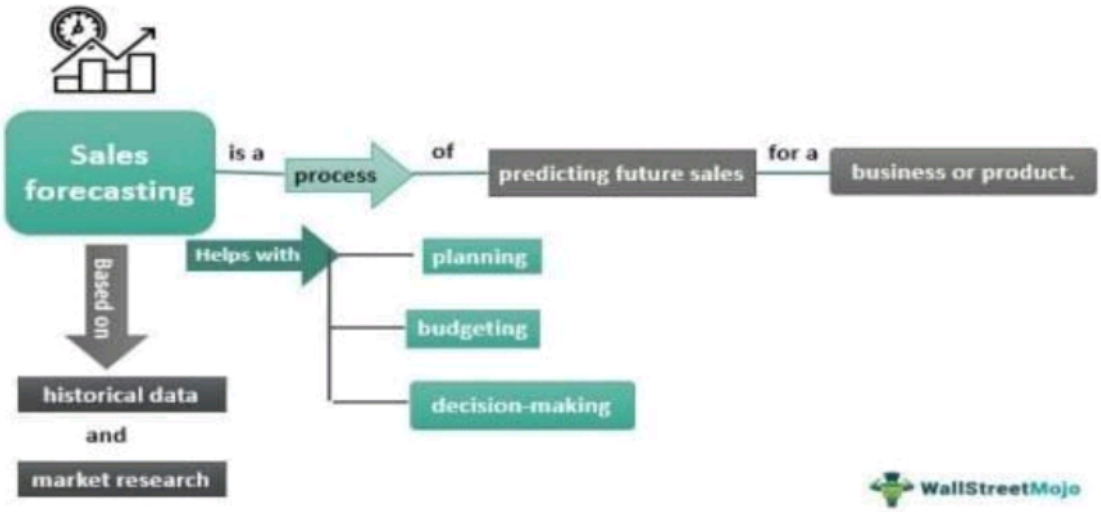
What is Sales Forecasting?

"Prediction based on past sales performance and an analysis of expected market conditions."

- The true value in making a forecast is that it forces us to look at the future objectively.

Abstract:

Connected devices, sensors, and mobile apps make the retail sector a relevant testbed for big data tools and applications. We investigate how big data is, and can be used in retail operations. Based on our state-of-the-art literature review, we identify four themes for big data applications in retail logistics: availability, assortment, pricing, and layout planning. Our semi-structured interviews with retailers and academics suggest that historical sales data and loyalty schemes can be used to obtain customer insights for operational planning, but granular sales data can also benefit availability and assortment decisions. External data such as competitors' prices and weather conditions can be used for demand forecasting and pricing. However, the path to exploiting big data is not a bed of roses. Challenges include shortages of people with the right set of skills, the lack of support from suppliers, issues in IT integration, managerial concerns including information sharing and process integration, and physical capability of the supply chain to respond to real-time changes captured by big data. We propose a data maturity profile for retail businesses and highlight future research directions. Association Rules is one of the data mining techniques which is used for identifying the relation between one item to another. Creating the rule to generate the new knowledge is a must to determine the frequency of the appearance of the data on the item set so that it is easier to recognize the value of the percentage from each of the datum by using certain algorithms, for example apriori. This research discussed the comparison between market basket analysis by using apriori algorithm and market basket analysis without using algorithm in creating rule to generate the new knowledge. The indicator of comparison included concept, the process of creating the rule, and the achieved rule. The comparison revealed that both methods have the same concept, the different process of creating the rule, but the rule itself remains the same.



Introduction:

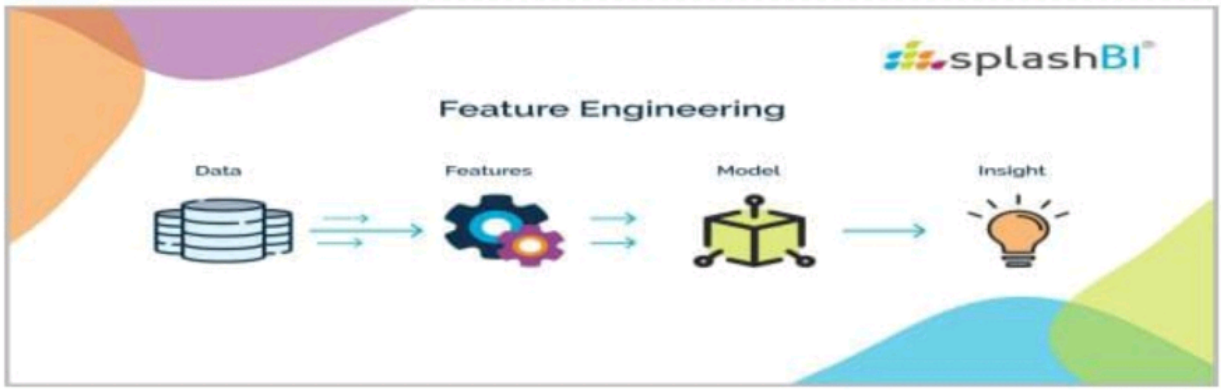
Future sales prediction is one of the most impactful machine learning use cases with broad applications across industries. Traditional future sales prediction techniques, such as ARIMA, often automatically incorporate the time component of the data by using lagged values of the target variable as model inputs. While these techniques provide interpretable coefficients that aid in understanding the contribution of each variable to the forecast, they can be sensitive to outliers, missing data, and changes in the underlying data-generating process over time. As a result, their accuracy may be compromised.


LIVE
TV

On the other hand, machine learning combined with feature engineering offers a more robust approach to future sales prediction. This approach can handle complex, non-linear relationships and is well-suited for large relational datasets with more complex relationships and intricate interdependencies.

Feature engineering plays a crucial role in future sales prediction, as it involves selecting and transforming raw data into meaningful features that can enhance the accuracy of predictive statistical models. The importance of feature engineering in time series modeling cannot be overstated.

By carefully selecting and transforming relevant features, statistical models become more adept at capturing the underlying patterns and relationships within the data. This ultimately leads to improved forecasting accuracy. Moreover, feature engineering enables the incorporation of domain knowledge and intuition, allowing the model to leverage human expertise and enhance performance.



Why is Feature Engineering Important?

Feature engineering is a critical step in building predictive models for demand forecasting. Here are some key reasons why it matters:

Enhanced Model Performance:

Well-engineered features can capture underlying patterns and relationships in the data, leading to more accurate predictions. By selecting the right features and transforming them appropriately, you can extract valuable information that might be hidden in the raw data.

Improved Interpretability:

Feature engineering can make your models more interpretable. By creating meaningful features, you can gain insights into which factors are driving demand and how they impact your predictions. This knowledge is invaluable for making informed business decisions.

Handling Non-linearity:

Real-world demand data is often non-linear, and feature engineering allows you to transform variables to better fit the assumptions of your chosen machine learning algorithm. This can result in better model performance.

Feature Engineering:

Feature engineering is the process of selecting and transforming relevant features from the raw data to improve the performance of ML models. In demand prediction for drugs on pharmacies, some of the most important features are:

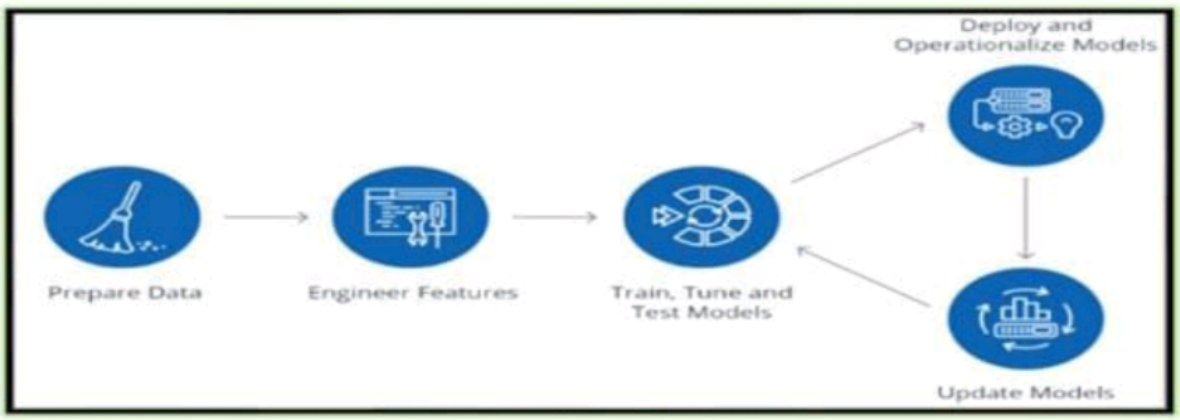
Time-based features:

These features capture trends and patterns over time. Examples include day of the week, month, year, and holidays.

Store-based features:

These features capture pharmacy-specific characteristics. Examples include the location of the pharmacy, the size of the pharmacy, and the customer demographics.

In addition to these features, external factors such as economic conditions, weather conditions, and cultural event can also be taken into account.



A typical approach to feature engineering in future sales prediction forecasting involves the following types of features:

Lagged variables:

By incorporating previous time series values as features, patterns such as seasonality and trends can be captured. For example, if we want to predict today's sales, using lagged variables like yesterday's sales can provide valuable information about the ongoing trend.

Moving window statistics:

This involves aggregating the time series values over a rolling window. By doing so, noise is smoothed out, shifting the focus to the underlying trends. Moving windows can help identify patterns that may not be immediately apparent in the raw data.

Time-based features:

Such as the day of the week, the month of the year, holiday indicators, seasonality, and other time-related patterns can be valuable for predictions. For instance, if certain products tend to have higher

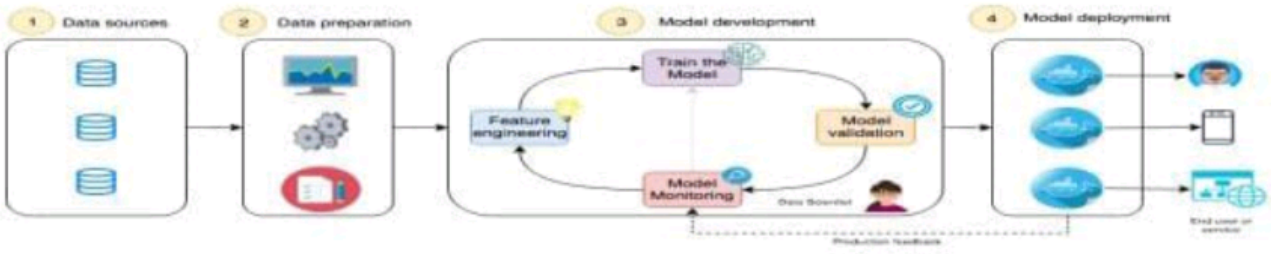

NEWS

average sales on weekends, incorporating the day of the week as a feature can improve the accuracy of the forecasting model.

Evaluation:

Once the model has been selected, it needs to be trained on the historical sales data. This involves splitting the data into training and testing sets, and using the training set to fit the model to the data. Once the model has been trained, it can be used to predict the demand for drugs in the pharmacy for a given period of time.

To evaluate the performance of the model, we can use metrics such as mean absolute percentage error (MAPE), root mean squared error (RMSE), mean absolute error (MAE), or R-squared(R^2). These metrics provide a measure of how well the model is able to predict the demand for drugs in the pharmacy. To evaluate the impact of feature engineering, you should compare the performance of models with and without engineered features using the evaluation metrics mentioned earlier. In most cases, you will likely observe that models with feature engineering outperform those without. Lower MAPE, RMSE, and MAE values and higher R^2 values are indicative of improved predictive accuracy.



To evaluate the given data set for future sales prediction, you would typically need to perform a series of steps. Here's a general outline of the evaluation process:


NEWS


TV MARKETING

To train a model using the given dataset, you can follow these general steps:

1.Data Exploration: Start by exploring the dataset to understand its structure, features, and target variable. This will help you gain insights into the data and make informed decisions during the modeling process.

2.Data Preprocessing: Clean and preprocess the data to handle missing values, outliers, and categorical variables. This may involve techniques such as imputation, scaling, encoding, etc.

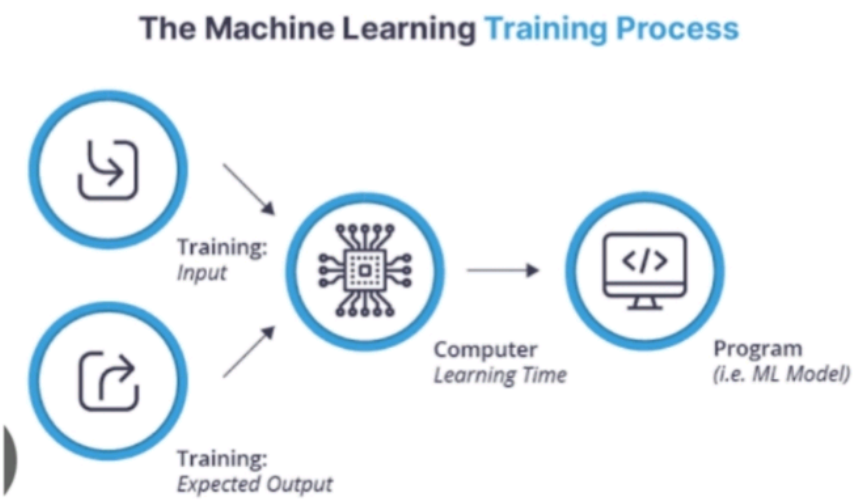
3.Feature Engineering: Create new features or transform existing ones to improve the predictive power of your model. This can include techniques like one-hot encoding, feature scaling, dimensionality reduction, etc.

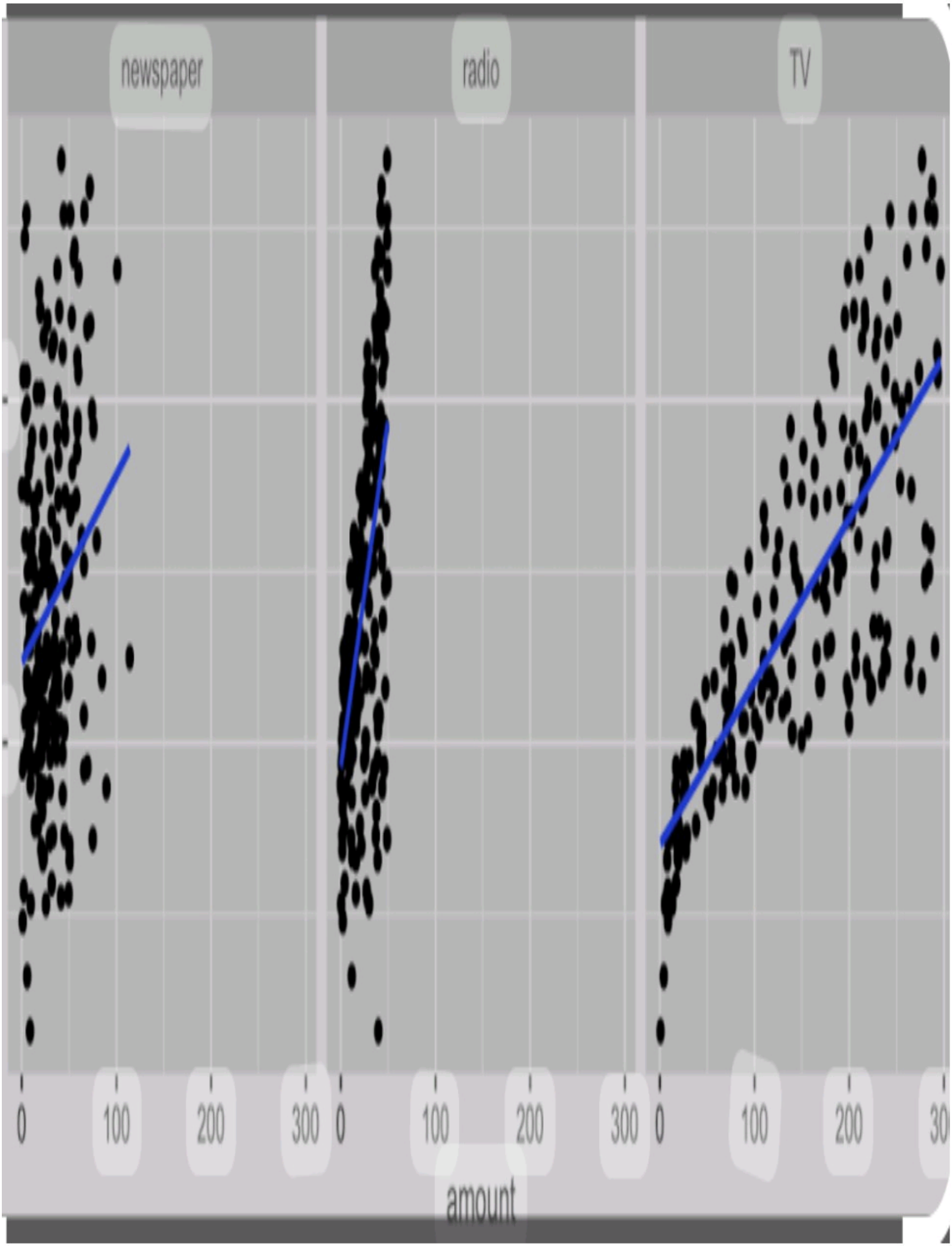
4.Model Selection: Choose an appropriate machine learning algorithm based on your problem type (classification or regression) and requirements. Some popular algorithms for regression tasks include linear regression, decision trees, random forests, gradient boosting methods (e.g., XGBoost), etc.

5.Model Training: Split your dataset into training and validation sets. Use the training set to train your chosen model using appropriate hyperparameters. Evaluate the model's performance on the validation set using suitable evaluation metrics (e.g., mean squared error for regression).

6.Model Evaluation: Assess how well your trained model performs on unseen data by evaluating it on a separate test set or using cross-validation techniques. Compare different models if necessary and select the best-performing one.

7.Hyperparameter Tuning: Fine-tune your chosen model by adjusting its hyperparameters to optimize its performance further. This can be done using techniques like grid search or random search.





Statics of sales for training dataset model

Data Preprocessing for ML

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split

# Step 1: Load the dataset
data = pd.read_csv("https://www.kaggle.com/datasets/chakradharmattapalli/future-sales-prediction")

# Step 2: Explore the dataset
print(data.head())
print(data.info())
print(data.describe())

# Step 3: Handle missing values (if any)
data.dropna(inplace=True)  # Drop rows with missing values

# Step 4: Extract relevant features
features = ['product_price', 'product_category', 'location', 'sale_date']

# Step 5: Create new features (if needed)
data['sale_month'] = pd.to_datetime(data['sale_date']).dt.month
data['sale_year'] = pd.to_datetime(data['sale_date']).dt.year

# Step 6: Encode categorical variables
encoder = OneHotEncoder(sparse=False)
encoded_features = encoder.fit_transform(data[['product_category', 'location']])
encoded_df = pd.DataFrame(encoded_features, columns=encoder.get_feature_names(['product_category',
'location']))
data_encoded = pd.concat([data, encoded_df], axis=1)

# Step 7: Scale numerical variables (if needed)
scaler = StandardScaler()
scaled_features = scaler.fit_transform(data_encoded[['product_price']])
scaled_df = pd.DataFrame(scaled_features, columns=['scaled_product_price'])
data_scaled = pd.concat([data_encoded, scaled_df], axis=1)

# Step 8: Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data_scaled.drop('future_sales', axis=1),
data_scaled['future_sales'], test_size=0.2, random_state=42)

# Step 9: Perform feature selection (if needed)
```

```
# Step 10: Save preprocessed data
X_train.to_csv('preprocessed_train_data.csv', index=False)
X_test.to_csv('preprocessed_test_data.csv', index=False)
```

To be a data preprocessing pipeline for a machine learning project. It follows the common steps in preparing a dataset for predictive modeling. Here's a brief explanation of each step:

We load the dataset from a URL using Pandas.
explore the dataset using head(), info(), and describe() to get an initial understanding of the data.
remove rows with missing values using dropna().
define a list of relevant features.
create new features, 'sale_month' and 'sale_year', from the 'sale_date' column.
use one-hot encoding to handle categorical variables like 'product_category' and 'location'.
We scale numerical variables (in this case, 'product_price') using StandardScaler.
We split the data into training and testing sets using train_test_split().
We mention the possibility of performing feature selection, which is a crucial step to choose the most relevant features for our model.
We can save the preprocessed data into separate CSV files for training and testing.
Keep in mind that for step 9, we may need to perform feature selection based on model performance and feature importance analysis. We could use techniques like feature importance from tree-based models or recursive feature elimination.

Additionally, ensure that the dataset URL is correct and accessible, and that we have the required libraries (e.g., Pandas, Scikit-learn) installed.

The code provided is a data preprocessing script, and it doesn't generate specific output that can be displayed here. However, I can explain what kind of output you might expect at each step in the code:

Loading the Dataset:

If the dataset is successfully loaded from the provided URL, we won't see any output.
Exploring the Dataset:

The print(data.head()) command will display the first few rows of the dataset.
print(data.info()) will show information about the dataset, including data types and non-null counts.
print(data.describe()) will provide summary statistics of the numerical columns.
Handling Missing Values:

No specific output is generated, but this step removes rows with missing values from the dataset.
Extracting Relevant Features:

No specific output is generated, but we create a list of relevant features.

Creating New Features:

No specific output is generated, but we add two new columns, 'sale_month' and 'sale_year'.
Encoding Categorical Variables:

This step might not generate any visible output, but we encode categorical variables and create a one-hot encoded DataFrame.
Scaling Numerical Variables:

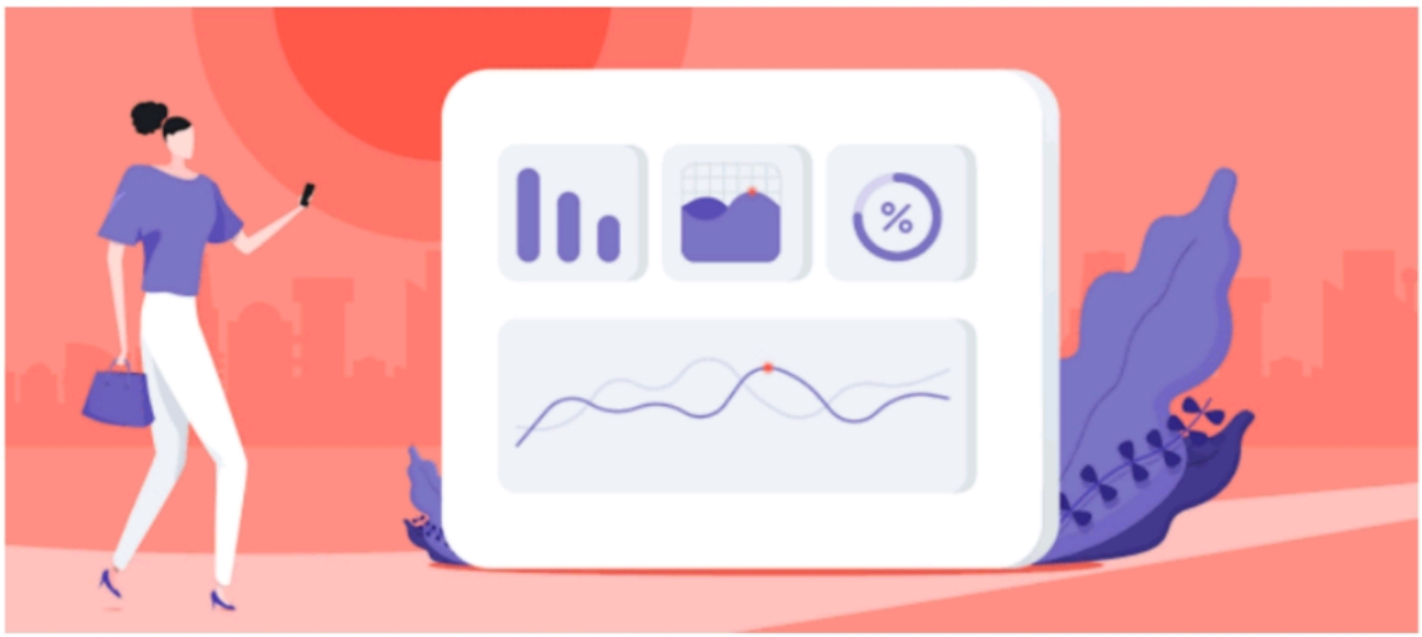
No specific output is generated, but we scale the 'product_price' column.
Splitting Data:

No specific output is generated, but we split the data into training and testing sets.
Feature Selection:

No output is provided for this step, as it's mentioned as a possibility, and the code doesn't perform feature selection.
Saving Preprocessed Data:

This step saves the preprocessed training and testing data as CSV files, 'preprocessed_train_data.csv' and 'preprocessed_test_data.csv', respectively.
The actual output or results depend on the contents of the dataset and whether there are any issues with the dataset loading or preprocessing steps. We can check the CSV files saved in step 10 to inspect the preprocessed data.


SALES PREDICTION




NEWS
NEWS

Conclusion

In the world of demand prediction, effective feature engineering is the key to building accurate and actionable predictive models. By understanding the importance of feature selection, transformation, and engineering techniques, data scientists and analysts can unlock the true potential of their data, enabling organizations to make informed decisions and stay ahead in a competitive marketplace.

To help you make the most out of your feature engineering efforts in demand prediction modeling, consider the following specific suggestions:

1.Domain Knowledge:

Invest time in understanding the domain you are working in. Speak with domain experts and stakeholders to identify which features are likely to have the most significant impact on demand. Their insights can guide your feature selection process.

2.Exploratory Data Analysis (EDA):

Conduct a thorough EDA to explore the relationships between your features and the target variable (demand). Visualizations, correlation analysis, and statistical tests can help you uncover important patterns and dependencies.

3.Feature Importance:

Utilize techniques like feature importance scores from tree-based models or feature selection algorithms to objectively assess the relevance of each feature. Focus on those with the highest importance scores.

4.Dimensionality Reduction:

If you have a large number of features, consider dimensionality reduction techniques like Principal Component Analysis (PCA) or t-Distributed Stochastic Neighbor Embedding (t-SNE) to reduce the feature space while preserving relevant information.

5.Feature Engineering Experiments:

Experiment with feature engineering techniques such as creating interaction features, polynomial features, or time-based aggregations. Test the impact of these engineered features on model performance.

THANK YOU