FUTURE SALES PREDICTION

PROBLEM DEFINITION AND DESIGN THINKING

PROBLEM STATEMENT:

The problem at hand is to develop a predictive model that leverages historical sales data to forecast future sales for a retail company. The primary objective of this project is to create a tool that empowers the company to optimize inventory management and make data-driven decisions. To achieve this, we need to perform data preprocessing, feature engineering, model selection, training, and evaluation.

DESIGN THINKING:

DATA SOURCE:

The first step is to utilize a dataset containing historical sales data. This dataset should include essential features such as date, product ID, store ID, and sales quantity. This data will serve as the foundation for our predictive model.

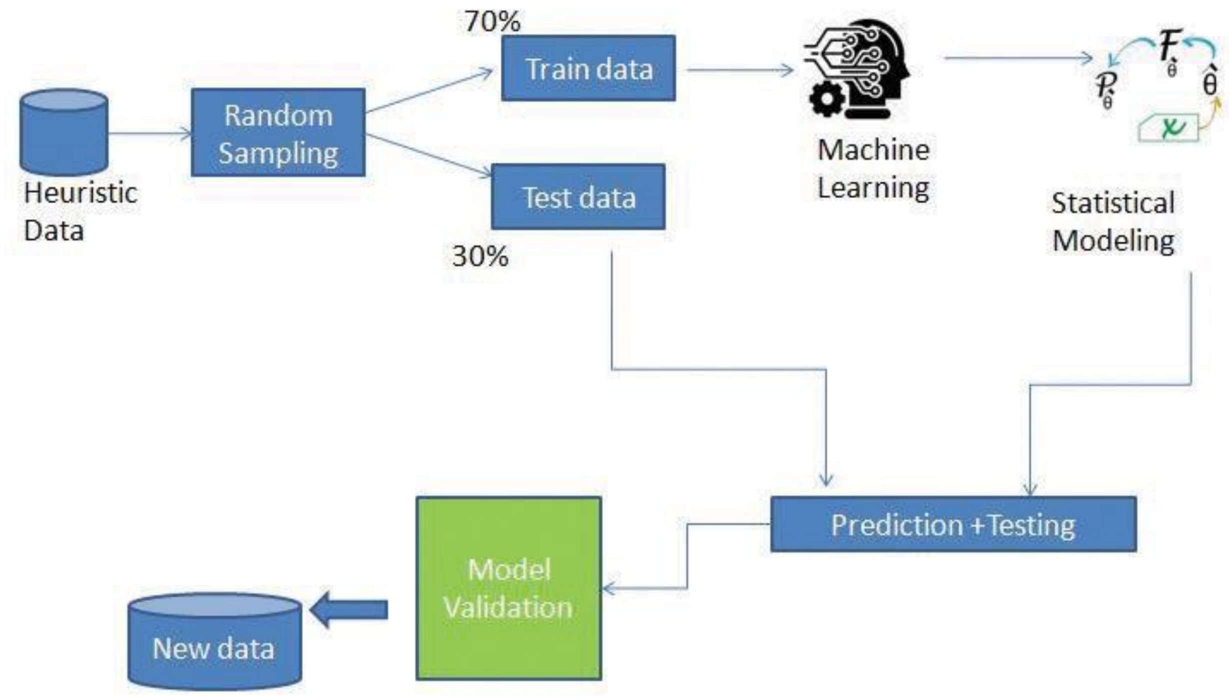
DATASET LINK : https://www.kaggle.com/datasets/chakradharmattapalli/future-sales-prediction

DATA PREPROCESSING:

Objective: Clean and preprocess the data to make it suitable for modelling.

STEPS:

1. Handle Missing Values: Identify and handle any missing or null values in the dataset. Depending on the extent of missing data, we may choose to impute values or drop rows/columns.
2. Convert Categorical Features: Convert categorical features like product ID and store ID into numerical representations using techniques such as one-hot encoding or label encoding.
3. Date Parsing: Extract relevant information from the date column, such as day of the week, month, or year. These time-based features can be valuable for forecasting.



FEATURE ENGINEERING:

Objective: Create additional features that can enhance the predictive power of the model.

STEPS:

1. Time-Based Features: Generate features such as day of the week, month, and year from the date column. These features can capture seasonality and trends.
2. Lag Features: Create lag features, where the target variable (sales quantity) from previous time steps is included as a feature. This can help capture autocorrelation in the data.

GOAL OF FEATURE ENGINEERING:

The goal of feature engineering is simply to make your data better suited to the problem at hand. Consider "apparent temperature" measures like the heat index and the wind chill. These quantities attempt to measure the perceived temperature to humans based on air temperature, humidity, and wind speed, things which we can measure directly. Feature engineering fulfills mainly two goals:
It prepares the input dataset in the form which is required for a specific model or machine learning algorithm.
Feature engineering helps in improving the performance of machine learning models magically.

MODEL SELECTION:

Objective: Choose suitable time series forecasting algorithms for predicting future sales.

STEPS:

1. Initial Model Selection: Start with basic time series models like Moving Averages to establish a baseline. Then, explore more advanced models like ARIMA (Auto Regressive Integrated Moving Average), Exponential Smoothing, or Prophet.
2. Hyperparameter Tuning: Fine-tune the hyperparameters of selected models for optimal performance. This may involve grid search or Bayesian optimization.

MODEL TRAINING:

Objective: Train the selected model using the pre-processed data.

STEPS:

1. Split the dataset into training and validation sets to assess model performance during training.
2. Train the model on the training data, taking into account the time-based structure of the data.

EVALUATION:

Evaluating Future Sales Predictions:

Testing the Model:

After selecting the best model, apply it to the testing set, which contains data that the model has not seen during training or validation. Evaluate the model's performance on the testing set using the same metrics used during validation. This step helps ensure that the model generalizes well to unseen data.

Cross-Validation:

In cases with limited data, use techniques like k-fold cross-validation to assess the model’s robustness and minimize overfitting.

Visualizations:

Create visualizations, such as time series plots, scatterplots, or residual plots, to gain insights into how the model is performing and where it might be falling short.

Residual Analysis:

Examine the residuals (differences between predicted and actual values) to identify any patterns or systematic errors in the model. This can help refine the model or identify potential improvements.

Business Impact Analysis:

Consider the practical impact of the model's predictions on the business. Understand how these predictions can inform strategic decisions, marketing campaigns, inventory management, and other areas of the business.

Model Maintenance:

Regularly update and retrain the model with new data to ensure it remains accurate over time. Markets change, and customer behaviour evolves, so the model needs to adapt.

Feedback Loop:

Establish a feedback loop to continuously learn from the model's predictions and refine the modelling process based on real-world outcomes.

Additional Metrics: Depending on the specific goals, consider other metrics like customer retention, customer acquisition cost, and lifetime value to assess the holistic impact of sales predictions on the business.

A/B Testing: Conduct A/B tests to validate the effectiveness of strategies and actions informed by the predictions.

PROCEDURE:

Step 1: The required libraries are imported.

Step 2: The given dataset is loaded in the program.

Step 3: To check whether the uploaded dataset is correct, a sample data is displayed in the output.

Step 4: The information from the dataset is summarized for better understanding.

Step 5: Then preprocessing is done, in which missing data, duplicate data are checked.

Step 6: If any modify the data or print False.

Step7:Then the features are extracted from the given data sheet.

Step 8:Split the data into training and testing sets.

Step 9:Initialize the regression algorithm.

Step 10:The set is trained and evaluated.

Step 11:The model is completed.

PROGRAM:

```
 # loading the data
import pandas as pd
if = pd.read_csv('/kaggle/input/future-sales-prediction/Sales.csv';
df.head();
df.shape;
df.info();
df.describe().T;
#Data processing(Outlier detection- using tv)
import seaborn as sns;
import matlablib. Pyplot as plt;
plt.figure(figsize=(8, 6));
sns.boxplot (x='Radio', data=df, palette='Oranges');
plt.title('Box Plot of Radio Advertising');
plt.xlabel('Radio Advertising Spending');
plt.grid(axis='x', linestyle='--', alpha=0.6);
# Show the plot
plt.show();
#exploratory data analysis (eda):
import plotly.express as px;
figure = px.scatter(df, x='Sales', y='TV', size='TV', trendline='ols', titl
e='Relationship Between Sales and TV Advertising');
figure.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGre
y')), selector=dict(mode='markers'));
figure.update_layout(
xaxis _title='Sales
yaxis_title='TV Advertising
legend_title='TV Ad Size'
plot_bgcolor='white';
figure.show()
```

```
#modeling and evaluation
# Linear Regression
from sklearn.linear_model import LinearRegression, Ridge, Lasso
linear_model = LinearRegression()
linear_mse, linear_rmse, linear_mae, linear_r2 = perform_cross_validation(linear_model, X, y,
num_folds)
print("Linear Regression:")
print(f"Average MSE: {np.mean(linear_mse) / np.mean(y) * 100:.2f}%")
print(f"Average RMSE: {np.mean(linear_rmse) / np.mean(y) * 100:.2f}%")
print(f"Average MAE: {np.mean(linear_mae) / np.mean(y) * 100:.2f}%")
print(f"Average R-squared: {np.mean(linear_r2) * 100:.2f}%")
print("\n")

# Ridge Regression
ridge_model = Ridge(alpha=1.0) # You can adjust alpha as needed
ridge_mse, ridge_rmse, ridge_mae, ridge_r2 = perform_cross_validation(ridge_model, X, y, num_folds)
print("Ridge Regression:")
print(f"Average MSE: {np.mean(ridge_mse) / np.mean(y) * 100:.2f}%")
print(f"Average RMSE: {np.mean(ridge_rmse) / np.mean(y) * 100:.2f}%")
print(f"Average MAE: {np.mean(ridge_mae) / np.mean(y) * 100:.2f}%")
print(f"Average R-squared: {np.mean(ridge_r2) * 100:.2f}%")
print("\n")

# Lasso Regression
lasso_model = Lasso(alpha=1.0) # You can adjust alpha as needed
lasso_mse, lasso_rmse, lasso_mae, lasso_r2 = perform_cross_validation(lasso_model, X, y, num_folds)
print("Lasso Regression:")
print(f"Average MSE: {np.mean(lasso_mse) / np.mean(y) * 100:.2f}%")
print(f"Average RMSE: {np.mean(lasso_rmse) / np.mean(y) * 100:.2f}%")
print(f"Average MAE: {np.mean(lasso_mae) / np.mean(y) * 100:.2f}%")
print(f"Average R-squared: {np.mean(lasso_r2) * 100:.2f}%")
print("\n")
# Decision Trees
from sklearn.tree import DecisionTreeRegressor
tree_model = DecisionTreeRegressor(max_depth=None, random_state=0) # You can adjust parameters as
needed
tree_mse, tree_rmse, tree_mae, tree_r2 = perform_cross_validation(tree_model, X, y, num_folds)
print("Decision Trees:")
print(f"Average MSE: {np.mean(tree_mse) / np.mean(y) * 100:.2f}%")
print(f"Average RMSE: {np.mean(tree_rmse) / np.mean(y) * 100:.2f}%")
print(f"Average MAE: {np.mean(tree_mae) / np.mean(y) * 100:.2f}%")
print(f"Average R-squared: {np.mean(tree_r2) * 100:.2f}%")
print("\n")

# Random Forest
from sklearn.ensemble import RandomForestRegressor
forest_model = RandomForestRegressor(n_estimators=100, random_state=0) # You can adjust
parameters as needed
forest_mse, forest_rmse, forest_mae, forest_r2 = perform_cross_validation(forest_model, X, y,
num_folds)
```

OUTPUT:
LOADING THE DATA

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

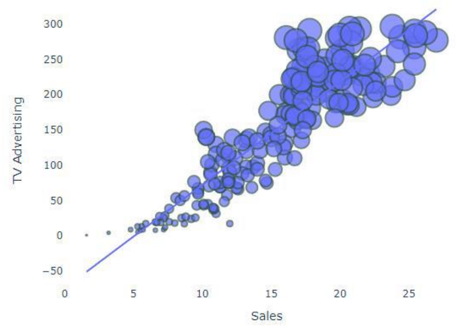
Shape-->(200, 4)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
# Column Non-Null Count Dtype
--- ------ -------------- -----
0 TV 200 non-null float64
1 Radio 200 non-null float64
2 Newspaper 200 non-null float64
3 Sales 200 non-null float64
dtypes: float64(4)
memory usage: 6.4 KB
```

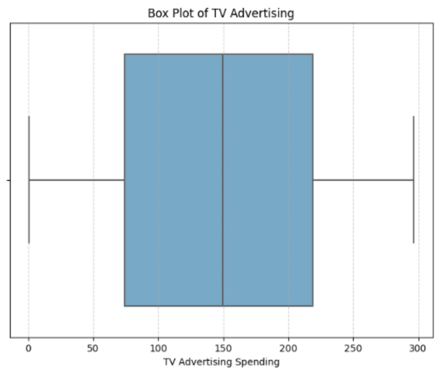
	count	mean	std	min	25%	50%	75%	max
TV	200.0	147.0425	85.854236	0.7	74.375	149.75	218.825	296.4
Radio	200.0	23.2640	14.846809	0.0	9.975	22.90	36.525	49.6
Newspaper	200.0	30.5540	21.778621	0.3	12.750	25.75	45.100	114.0
Sales	200.0	15.1305	5.283892	1.6	11.000	16.00	19.050	27.0

```
Linear Regression:

Average MSE: 18.90%
Average RMSE: 11.01%
Average MAE: 8.38%
Average R-squared: 89.53

Ridge Regression:
Average MSE: 19.67%
Average RMSE: 11.20%
Average MAE: 8.54%
Average R-squared: 89.19%

Lasso Regression:
Average MSE: 115.55%
Average RMSE: 27.51%
Average MAE: 22.39%
Average R-squared: 35.98%

Decision Trees:
Average MSE: 16.73%
Average RMSE: 10.40%
Average MAE: 7.56%
Average R-squared: 90.65%
```

Relationship between sales and tv advertising:



#Data processing(Outlier detection- using tv)



Conclusion:

This document outlines the plan for developing a predictive model to forecast future sales for a retail company. The process involves data preprocessing, feature engineering, model selection, training, and evaluation. By following this structured approach, we aim to provide the company with a powerful tool for optimizing inventory management and making informed business decisions based on data-driven sales predictions.