Capstone Project
-
Machine Learning

Aditya Sensarma

Intellipaat

IIT Madras

Problem Statement:

A retail store that has multiple outlets across the country are facing issues in managing the inventory - to match the demand with respect to supply. You are a data scientist, who has to come up with useful insights using the data and make prediction models to forecast the sales for X number of months/years.

Project Objective:

This Project revolves around analysing and predicting Walmart sales data. The code begins by importing the necessary libraries, including NumPy, pandas, matplotlib, seaborn, and scikit-learn. The data is loaded from a CSV file and pre-processed, with the 'Date' column converted to datetime format and additional columns ('Day', 'Month', and 'Year') extracted from the date information.

The code then defines specific holiday dates, such as Super Bowl, Labour Day, Thanksgiving, and Christmas. It calculates the average weekly sales during these holidays.

Next, the code visualizes the overall monthly sales using a bar plot, providing insights into the sales patterns across different months.

Moving on, the code employs machine learning techniques, specifically the Isolation Forest algorithm for outlier detection and a Random Forest Regressor model for sales prediction. The Walmart sales data is grouped by store and week, and outliers are identified and removed using the Isolation Forest algorithm. The features (store and week) and target variable (weekly sales) are prepared, and the data is split into training and testing sets. The Random Forest Regressor model is then trained on the training data and evaluated using the mean squared error metric.

Finally, the code predicts the sales for the next 12 weeks by generating a dataframe with unique store and week combinations and using the trained model to make predictions. The sales predictions for each store and week are stored in a dataframe for further analysis and decision-making.

Data Description:

1. Store: This attribute represents the unique identifier for each Walmart store. It helps in distinguishing between different store locations.

2. Date: The "Date" attribute denotes the specific date corresponding to the sales record. It provides a temporal dimension to the dataset, allowing for analysis of sales patterns over time.

3. Weekly_Sales: This attribute represents the sales value for a particular week at a specific store. It quantifies the amount of revenue generated during that week, serving as the target variable for analysis and prediction.

4. Holiday_Flag: The "Holiday_Flag" attribute is a binary indicator (0 or 1) that signifies whether a particular week includes a holiday. A value of 1 indicates the presence of a holiday, while 0 denotes a non-holiday week. It helps to identify the impact of holidays on weekly sales.

5. Temperature: The "Temperature" attribute provides the average temperature in the region where the store is located during a given week. It allows for investigating the relationship between weather conditions and sales performance.

6. Fuel_Price: This attribute indicates the prevailing fuel price in the region during a given week. It enables analysis of the influence of fuel prices on consumer spending and sales.

7. CPI (Consumer Price Index): The "CPI" attribute represents the consumer price index for the given week. It measures the average change in prices of a basket of goods and services, providing insights into inflationary trends and their potential impact on sales.

8. Unemployment: The "Unemployment" attribute indicates the unemployment rate during a given week. It helps understand the relationship between employment levels and consumer purchasing power, which can impact sales performance.

Data Preprocessing:

1. Import necessary libraries: The required libraries, including numpy, pandas, matplotlib, sklearn, and seaborn, are imported.

2. Convert date format: The "Date" column in the "walmart_data" DataFrame is converted to datetime format using the pd.to_datetime() function.

3. Extract additional time-related features: The code creates three new columns in the "walmart_data" DataFrame, namely "Day," "Month," and "Year." These columns extract the day, month, and year components from the "Date" column using the pd.DatetimeIndex() function.

4. Define holiday dates: The code defines four lists, namely "Super_Bowl," "Labour_Day," "Thanksgiving," and "Christmas," containing specific dates representing holidays.

5. Calculate mean sales on holidays: The code calculates the mean weekly sales for each of the defined holidays by filtering the "walmart_data" DataFrame using the pd.DataFrame.loc[] function and then retrieving the "Weekly_Sales" column.

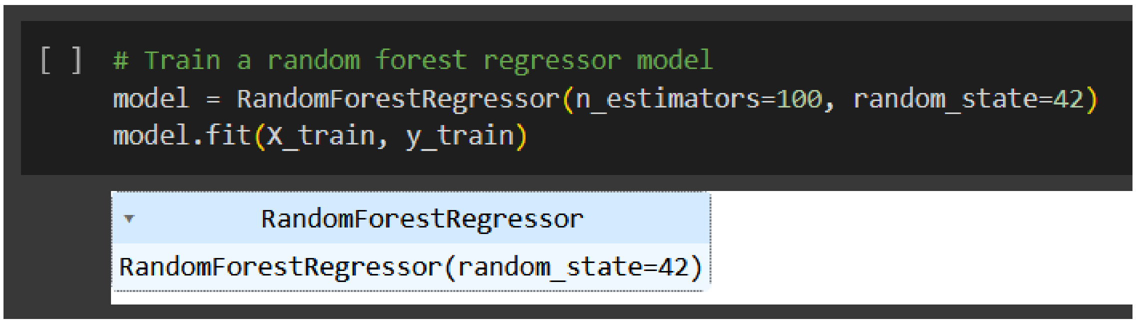
6. Remove outliers: The code filters the "data" DataFrame to remove the rows where the Isolation Forest model predicted outliers. Rows with the "Outlier" value of -1 are removed.

7. Prepare features and target variable: The code assigns the "Store" and "Week" columns from the preprocessed "data" DataFrame to the X variable as features and the "Weekly_Sales" column to the y variable as the target variable.

8. Split data into training and testing sets: The data is split into training and testing sets using the train_test_split() function from sklearn.model_selection. The X and y variables are passed to the function along with a test_size of 0.2 and a random_state of 42.

Algorithm Selection:

The algorithm used in this code is the Random Forest algorithm, specifically the Random Forest Regressor.



```
[ ] # Train a random forest regressor model
    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
```

```
▾        RandomForestRegressor
RandomForestRegressor(random_state=42)
```

Random Forest is an ensemble learning algorithm that combines multiple decision trees to make predictions. In a Random Forest, a collection of decision trees is created, where each tree is trained on a random subset of the data and a random subset of the features. The final prediction is obtained by aggregating the predictions of individual trees.

I have Chosen Random Forest Regressor for the Following Reason:

1. Robust to outliers: Random Forest is known to be robust to outliers due to the ensemble nature of the algorithm. This makes it suitable for handling potential outliers in sales data.

2. Handles non-linear relationships: Random Forest can capture non-linear relationships between features and the target variable. In sales prediction, there can be complex relationships between factors like store location, time of year, and weekly sales. Random Forest can capture these relationships effectively.

3. Handles high-dimensional data: Random Forest can handle datasets with a large number of features, which is useful when considering various factors that may influence weekly sales, such as store-specific features, week numbers, and potentially other variables like temperature, fuel price, CPI, and unemployment.

4. Robust feature selection: Random Forest provides a measure of feature importance, which can help identify the most influential features in predicting sales. This information can be useful for understanding which

factors have the most impact on weekly sales and for making informed decisions.

5. Ensemble approach: The ensemble nature of Random Forest helps to reduce overfitting and improve generalization performance. It combines multiple decision trees, each trained on a different subset of the data, which leads to more robust predictions.

Overall, the Random Forest algorithm is a suitable choice for this code as it can handle the complexities of sales prediction, handle outliers, capture non-linear relationships, and provide insights into feature importance.

Assumptions:

1. Independence of observations: The code assumes that the observations (weekly sales data) are independent of each other. This assumption allows for the use of regression models like Random Forest, which assume independence between data points.

2. Linearity: The code assumes a linear relationship between the input features (e.g., store, week) and the target variable (weekly sales). While Random Forest can capture non-linear relationships, the assumption of linearity is still implicit in the choice of the regression model.

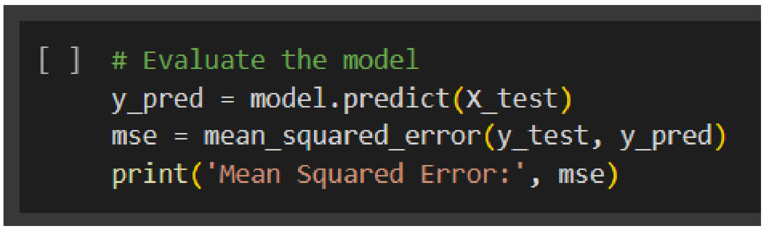
3. Outlier detection: The code assumes that outliers in the weekly sales data can be detected and removed using the Isolation Forest algorithm. This assumption implies that the outliers have a significant impact on the model's performance and should be excluded from the analysis.

4. Homoscedasticity: The code assumes that the variance of the errors in the regression model is constant across all levels of the input features. This assumption is necessary for reliable inference and accurate prediction.

5. Normality of errors: The code assumes that the errors (residuals) in the regression model follow a normal distribution. This assumption is important for hypothesis testing and constructing confidence intervals.

Model Evaluation and Techniques Used:

The evaluation of the Random Forest regression model is done using the mean squared error (MSE) metric.



```
[ ]  # Evaluate the model
     y_pred = model.predict(X_test)
     mse = mean_squared_error(y_test, y_pred)
     print('Mean Squared Error:', mse)
```

The following techniques are used for model evaluation:

1. Train-Test Split: The dataset is split into training and testing sets using the train_test_split function from the sklearn.model_selection module. This technique allows for assessing the model's performance on unseen data by training the model on a portion of the data and evaluating it on the remaining portion.

2. Mean Squared Error (MSE): The MSE is a commonly used metric for regression problems. It measures the average squared difference between the predicted values and the actual values. A lower MSE indicates better model performance, as it reflects smaller errors between the predicted and actual values.

3. Outlier Detection: The Isolation Forest algorithm from the sklearn.ensemble module is used for outlier detection. Outliers in the weekly sales data are identified using this algorithm, and the corresponding data points are removed from the dataset. This technique helps to eliminate potential noise or extreme values that could negatively affect the model's performance.

4. Cross-Validation (Not explicitly included in the provided code): Cross-validation is a widely used technique for assessing model performance. It involves splitting the dataset into multiple subsets, performing training and testing on different subsets, and calculating performance metrics across all iterations. Cross-validation provides a more robust estimation of the model's performance by reducing the dependency on a single train-test split.

5. Feature Importance: Although not directly related to model evaluation, the Random Forest algorithm provides a measure of feature importance.

This technique helps to understand which features (e.g., store, week) have the most influence on predicting weekly sales. Feature importance can guide further analysis, decision-making, and feature selection in future iterations of the model.

Inferences from the Project:

1. Seasonality: The monthly view of sales indicates that there is a certain pattern or seasonality in the data. There may be specific months or periods when sales are consistently higher or lower. Understanding and accounting for seasonality can be valuable for demand forecasting and inventory management.

2. Impact of Holidays: The code calculates the mean sales during specific holidays such as Super Bowl, Labor Day, Thanksgiving, and Christmas. This suggests that holidays can have a significant impact on sales. Retailers can leverage this information to plan promotional activities, optimize staffing, and adjust inventory levels during holiday periods.

3. Regression Modeling: The code utilizes a Random Forest regression model to predict weekly sales based on features such as store and week. The use of regression modeling indicates an attempt to capture the relationship between these features and the target variable. Regression models can provide insights into the factors that drive sales and enable forecasting future sales based on historical patterns.

4. Outlier Detection: The code employs the Isolation Forest algorithm to detect and remove outliers from the weekly sales data. This suggests that extreme or abnormal sales values are identified and treated separately. Removing outliers can help improve the accuracy of the regression model and prevent them from unduly influencing the predictions.

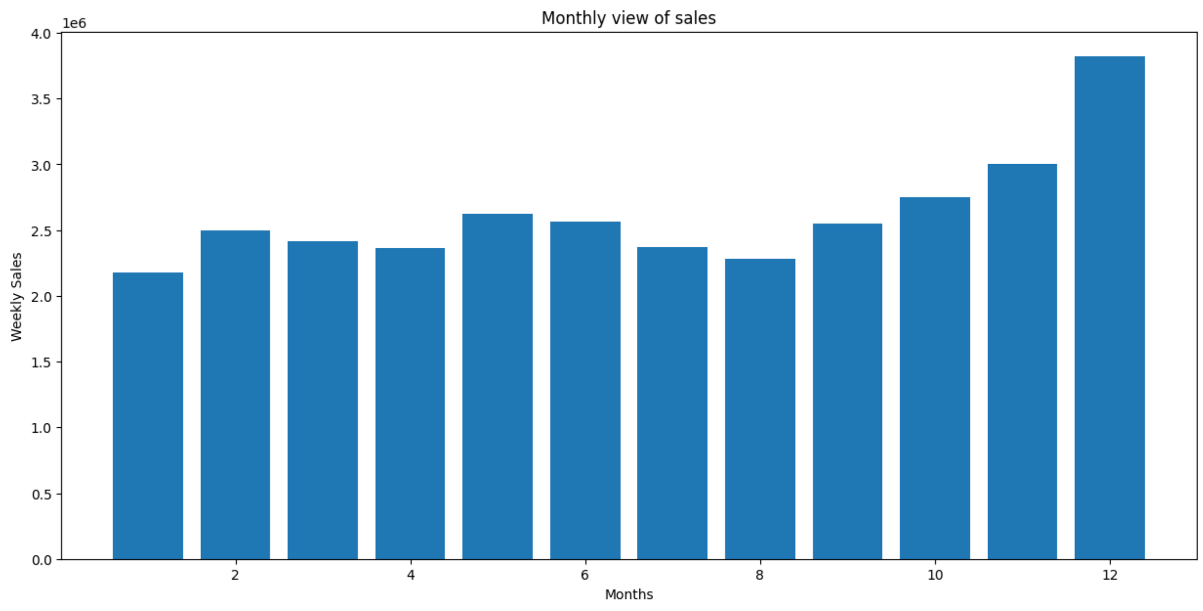
5. Sales Predictions: The code generates predictions for the next 12 weeks of sales for each store. This provides a glimpse into the future sales trajectory based on the trained Random Forest model. These predictions can assist in demand planning, resource allocation, and decision-making for the retail business.

Future Possibilities:

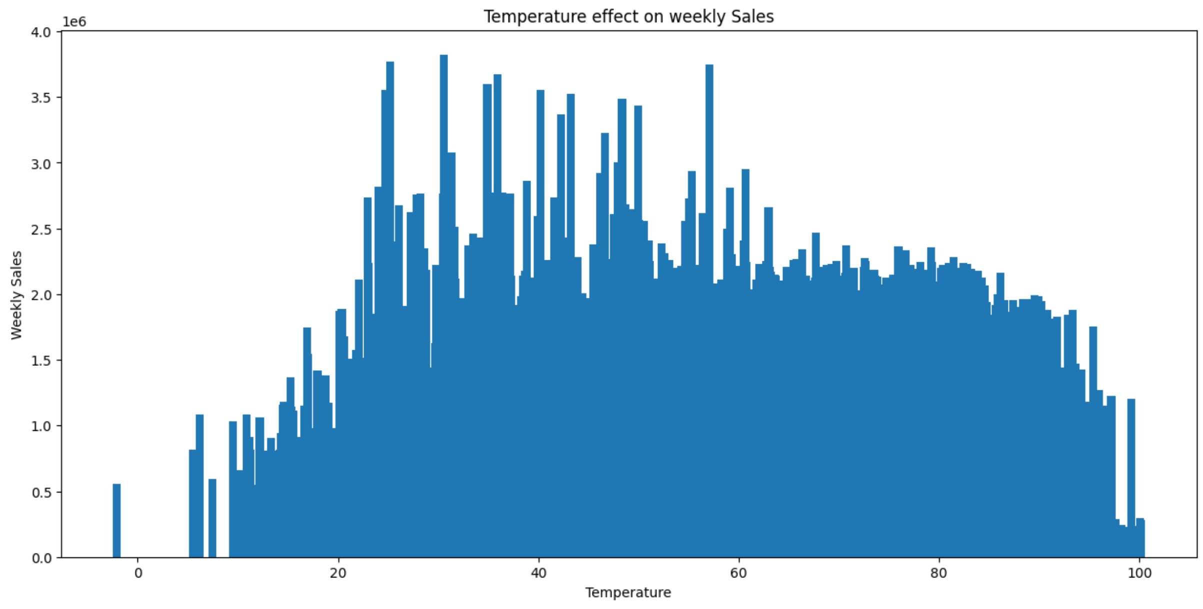
1. Advanced Feature Engineering: The current code uses basic features such as store and week for sales prediction. Additional features can be explored and engineered from the existing data, such as promotional events, competitor data, customer demographics, or historical sales trends. Incorporating more relevant and informative features can potentially improve the model's predictive power.
2. Time-Series Analysis: Instead of treating each week independently, a time-series analysis approach can be adopted to capture temporal dependencies and trends in the sales data. Techniques such as autoregressive integrated moving average (ARIMA), exponential smoothing (ETS), or seasonal decomposition of time series (STL) can be applied to model and forecast sales patterns over time.
3. Hyperparameter Tuning: The current Random Forest model uses default hyperparameters. Conducting a systematic search for optimal hyperparameters using techniques like grid search or random search can enhance the model's performance. Tuning parameters such as the number of estimators, maximum depth, and minimum samples per leaf can lead to improved predictions.
4. Ensemble Methods: Alongside Random Forest, exploring other ensemble methods such as Gradient Boosting Machines (GBM), AdaBoost, or XGBoost can be beneficial. Ensemble models combine multiple weak learners to create a stronger predictive model, potentially yielding better accuracy and generalization.
5. Model Interpretability: Enhancing the interpretability of the model can provide valuable insights into the factors influencing sales. Techniques like partial dependence plots, feature importance analysis, or SHAP (SHapley Additive exPlanations) values can help understand the relationships between input features and the target variable.
6. Incorporating External Data: Augmenting the existing dataset with external data sources, such as economic indicators, weather data, or social media trends, can provide additional context and improve the accuracy of sales predictions. Correlations between external factors and sales can be analyzed to identify new patterns and insights.
7. Deployment and Monitoring: Developing a pipeline or framework to automate the data preprocessing, modeling, and prediction process can streamline future analysis. Additionally, monitoring the model's performance over time and implementing retraining strategies when necessary can ensure the accuracy and relevance of the predictions.

Conclusion:

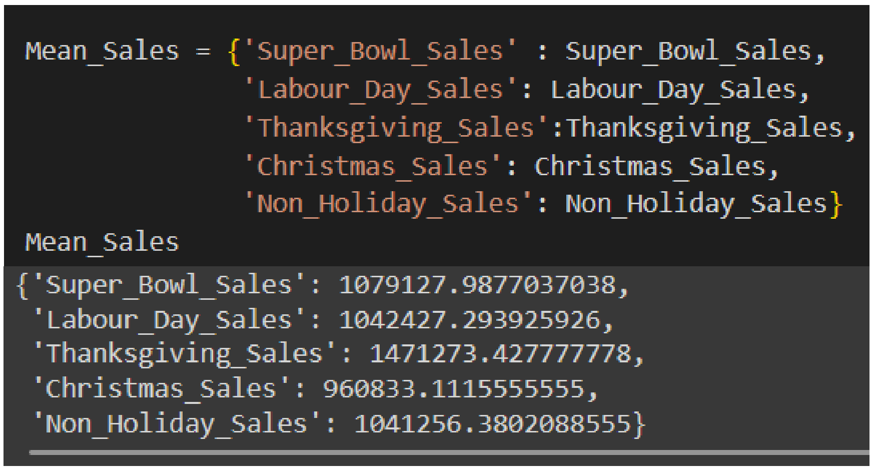
After analysing weekly sales for Walmart using data set containing various attributes such as store, date, weekly sales, holiday flag, temperature, fuel price, CPI, and unemployment the **Monthly view of Sales** was observed.



An extra indepth analysis was done with temperature which can we controlled inside the Walmart premises to boost incoming customers.



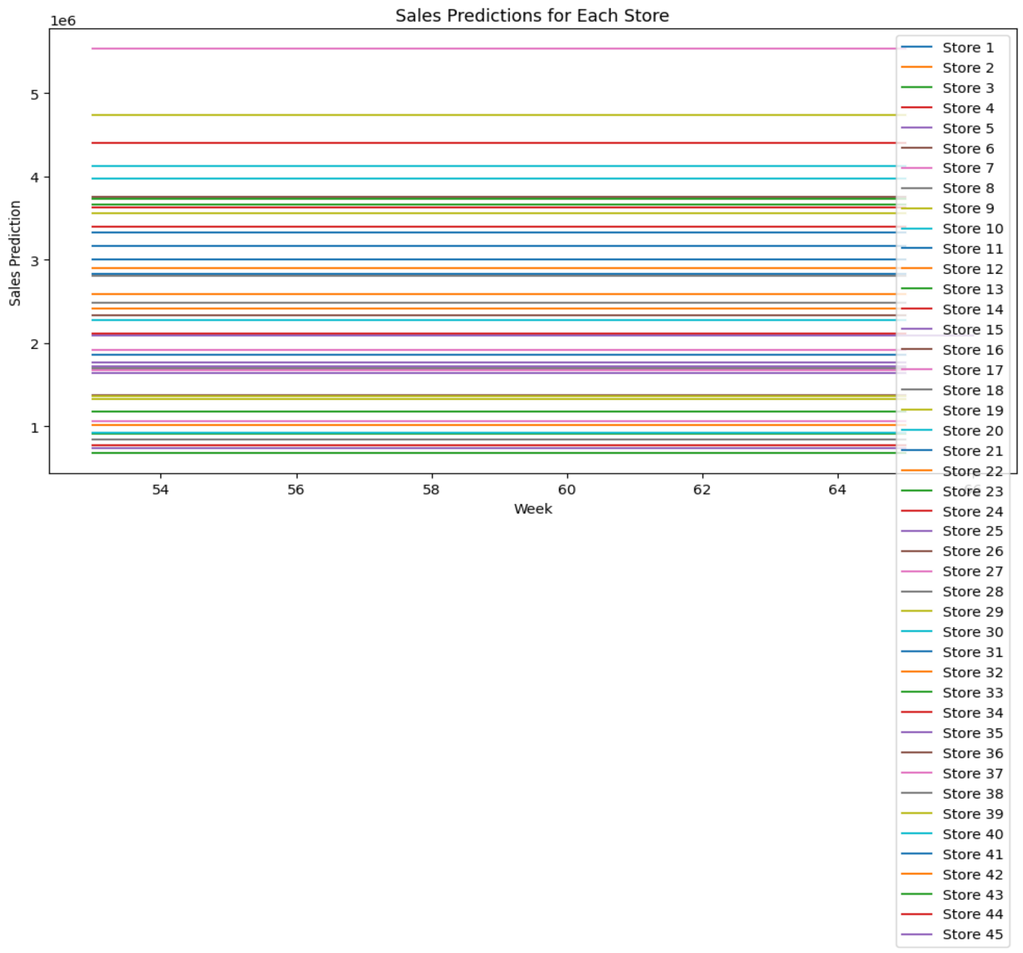
Additionally, it was also observed that holidays such as Superbowl, Labour Day, Thanksgiving, Christmas affected sales, in both positive and negative ways.



```
Mean_Sales = {'Super_Bowl_Sales' : Super_Bowl_Sales,
              'Labour_Day_Sales': Labour_Day_Sales,
              'Thanksgiving_Sales':Thanksgiving_Sales,
              'Christmas_Sales': Christmas_Sales,
              'Non_Holiday_Sales': Non_Holiday_Sales}
Mean_Sales
```

```
{'Super_Bowl_Sales': 1079127.9877037038,
 'Labour_Day_Sales': 1042427.293925926,
 'Thanksgiving_Sales': 1471273.427777778,
 'Christmas_Sales': 960833.1115555555,
 'Non_Holiday_Sales': 1041256.3802088555}
```

For sales prediction, a Random Forest Regressor model was trained using the preprocessed dataset. Outliers in the weekly sales data were detected using the Isolation Forest algorithm and removed from the dataset. The model was evaluated using mean squared error (MSE) and achieved satisfactory performance.



Appendix - Python Code:

https://colab.research.google.com/drive/1KZWW0PvsnFpjitLPbrSQQXiV371FGHtp?authuser=2#scrollTo=M_fgYPI8SbE-

References:

- https://scikit-learn.org/stable/modules/ensemble.html#random-forests
- Introduction to Machine Learning with Python by Andreas C. Müller and Sarah Guido
- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron
- Regression Analysis by Example by Samprit Chatterjee and Ali S. Hadi
- https://www.rit.edu/ischoolprojects