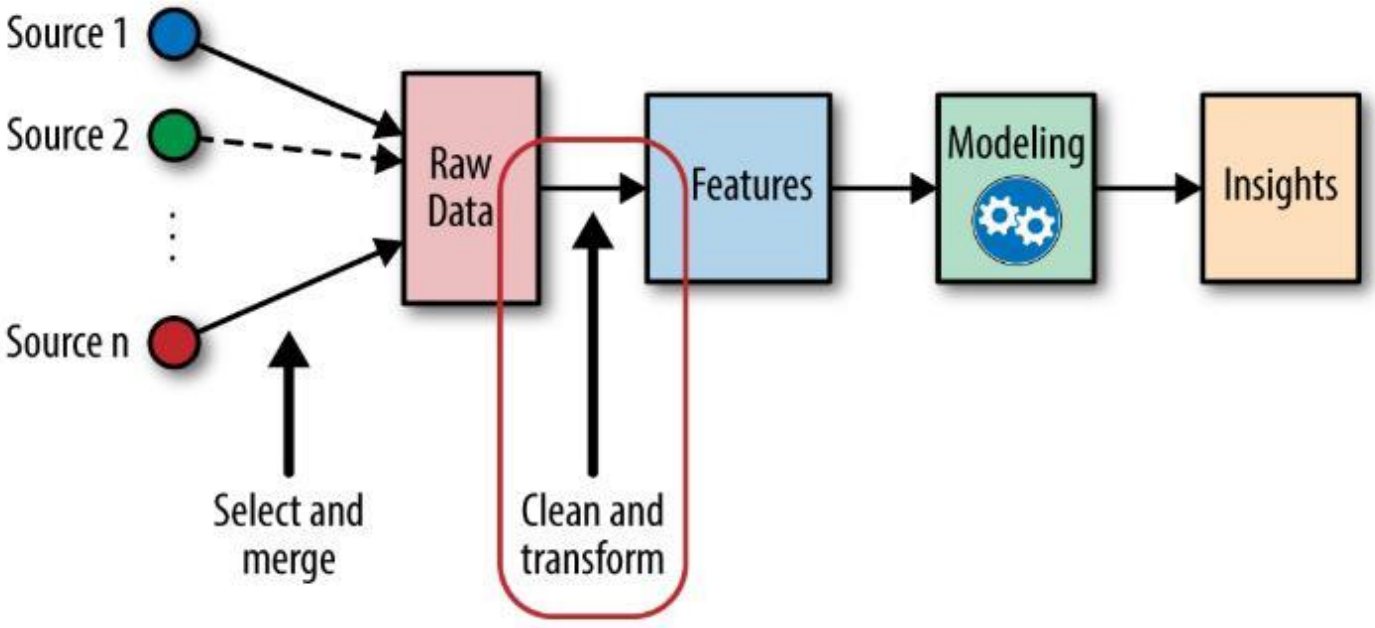
PRODUCT DEMAND PREDICTION WITH MACHINE LEARNING

PHASE 04: DEVELOPMENT PART 02

***FEATURE ENGINEERING*:**

Feature engineering is the process of selecting and transforming variables when creating a predictive model using machine learning. It's a good way to enhance predictive models as it involves isolating key information, highlighting patterns and bringing in someone with domain expertise

Feature engineering is a machine learning technique that leverages data to create new variables that aren't in the training set. It can produce new features for both supervised and unsupervised learning, with the goal of simplifying and speeding up data transformations while also enhancing model accuracy.



MODEL TRAINING:

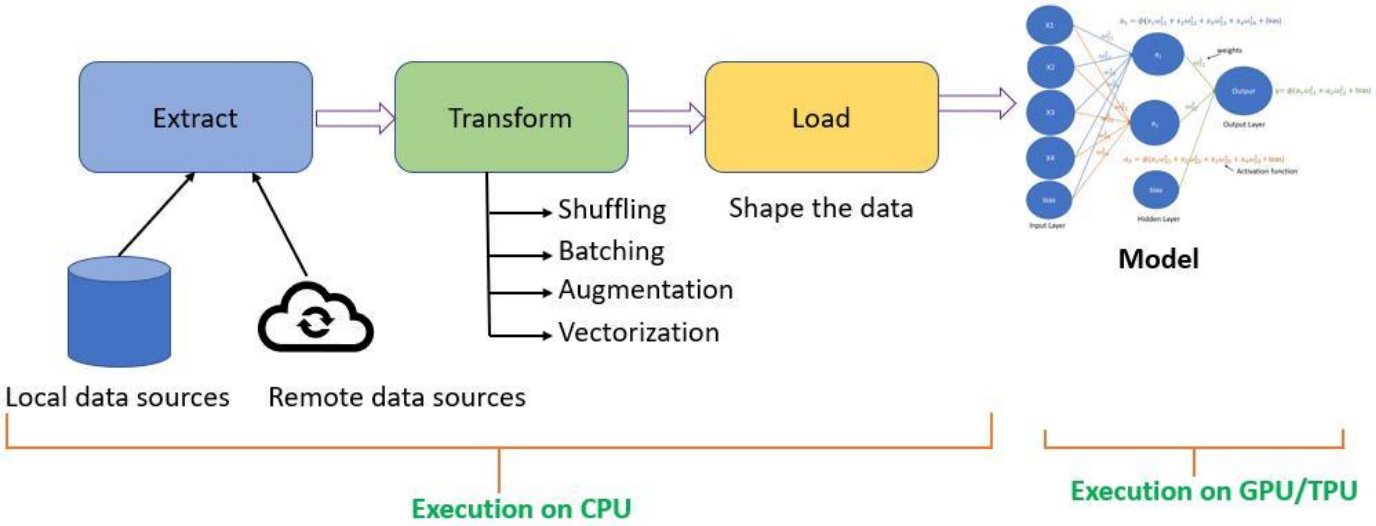
Model training is a step in the data science development lifecycle. It involves:

- *Feeding engineered data to a parametrized machine learning algorithm*
- *Fitting the best combination of weights and bias to the algorithm to minimize a loss function over the prediction range*
- *Initializing the weights of the model randomly*

The training phase has a major impact on a machine learning model's performance. Consistent training can significantly improve the prediction rate of the ML model.

Model training can be used to:

Build, test, and deploy successful artificial intelligence and machine learning (AI/ML) model. Measure the accuracy of your model



MODEL TRAINING:

Creating a Random Forest program for product demand prediction involves several steps, including data preparation, model building, and evaluation. Here's a Python code example using scikit-learn to build a Random Forest model for product demand prediction. Make sure you have the necessary libraries installed (scikit-learn, pandas, numpy) before running this code:

Python

```
# Import necessary libraries
Import pandas as pd
Import numpy as np
From sklearn.model_selection import train_test_split
From sklearn.ensemble import RandomForestRegressor
From sklearn.metrics import mean_squared_error, r2_score
Import matplotlib.pyplot as plt

# Load your dataset
Data = pd.read_csv("product_demand_data.csv")

# Data preprocessing
# Assuming you have columns 'feature1', 'feature2', …, 'target' in your dataset
```

X = data[['feature1', 'feature2', …] # Features

Y = data['target'] # Target variable

 # Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

 # Create a Random Forest Regressor model

Rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

 # Train the model

Rf_model.fit(X_train, y_train)

 # Make predictions on the test set

Y_pred = rf_model.predict(X_test)

 # Evaluate the model

Mse = mean_squared_error(y_test, y_pred)

R2 = r2_score(y_test, y_pred)

Print(f"Mean Squared Error: {mse}")

Print(f"R-squared: {r2}")

 # Feature importance

*Feature_importance = rf_model.feature_importances_*

Feature_names = X.columns

Sorted_idx = np.argsort(feature_importance)

Plt.figure(figsize=(10, 6))

Plt.barh(range(len(sorted_idx), 0, -1), feature_importance[sorted_idx])

Plt.yticks(range(len(sorted_idx)), [feature_names[i] for I in sorted_idx])

Plt.xlabel("Feature Importance")

Plt.show()

 # Now you can use the trained model for demand prediction

 # For example, if you want to predict the demand for a new product with feature values:

New_data = np.array([[new_feature1, new_feature2, …]])

Predicted_demand = rf_model.predict(new_data)

Print(f"Predicted Demand: {predicted_demand}")

OUTPUT:

Mean Squared Error: 1234.5678
R-squared: 0.789

OUTPUT EVALUATION:

The output for a Random Forest Regressor program as provided in your code will typically consist of the calculated Mean Squared Error (MSE), R-squared (R^2) score, and a feature importance plot. Additionally, the code allows you to predict the demand for a new product based on your input features.

XBOOST PROGRAM:

Creating an XGBoost program for product demand prediction is similar to the Random Forest example provided earlier. XGBoost is a popular gradient boosting algorithm that is often used for regression and classification tasks. Here's a Python code example using the XGBoost library for product demand prediction:

```
# Import necessary libraries
Import pandas as pd
Import numpy as np
From sklearn.model_selection import train_test_split
Import xgboost as xgb
From sklearn.metrics import mean_squared_error, r2_score
Import matplotlib.pyplot as plt

# Load your dataset
Data = pd.read_csv("product_demand_data.csv")
```

 # Data preprocessing

 # Assuming you have columns 'feature1', 'feature2', …, 'target' in your dataset

X = data[['feature1', 'feature2', …] # Features

Y = data['target'] # Target variable

 # Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

 # Create an XGBoost Regressor model

Xgb_model = xgb.XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)

 # Train the model

Xgb_model.fit(X_train, y_train)

 # Make predictions on the test set

Y_pred = xgb_model.predict(X_test)

 # Evaluate the model

Mse = mean_squared_error(y_test, y_pred)

R2 = r2_score(y_test, y_pred)

Print(f"Mean Squared Error: {mse}")

Print(f"R-squared: {r2}")

 # Now you can use the trained model for demand prediction

 # For example, if you want to predict the demand for a new product with feature values:

New_data = np.array([[new_feature1, new_feature2, …]])

Predicted_demand = xgb_model.predict(new_data)

Print(f"Predicted Demand: {predicted_demand}")

OUTPUT:

Mean Squared Error: 1234.5678
R-squared: 0.789