Project 10: Product Demand Prediction with Machine Learnings

Feature Engineering

Feature engineering is a crucial step in building effective machine learning models for production demand prediction. By selecting and transforming the right features, you can improve the model's accuracy and its ability to make meaningful predictions. Here are some feature engineering techniques and considerations for production demand prediction with machine learning:

Historical Data:

1. **Time-based features:** Incorporate time-related information such as day of the week, month, quarter, and year to capture seasonality and trends.
2. **Rolling statistics:** Calculate rolling averages, moving sums, or other statistics to capture short-term and long-term trends.
3. **Lag features:** Include lagged values of the target variable or other relevant features to account for temporal dependencies.

Categorical Variables:

1. **One-Hot Encoding:** Convert categorical variables into binary features using one-hot encoding.
2. **Target encoding:** Encode categorical variables based on the mean or median of the target variable for each category.
3. **Embeddings:** For high-cardinality categorical variables, consider using embeddings generated by neural networks.

External Data:

Incorporate external data sources such as weather data, holidays, or economic indicators that might impact demand.

Domain-Specific Features:

Create features that are specific to the production process or industry. For example, in manufacturing, you might consider equipment uptime, production line data, or inventory levels.

Text Data:

If you have text data related to products or customers, consider using natural language processing techniques to extract relevant information.

Aggregation:

Aggregate data at different levels of granularity, such as daily, weekly, or monthly, to capture patterns and trends.

Scaling and Normalization:

Standardize or normalize numerical features to ensure that they are on the same scale. This helps the machine learning algorithm perform better.

Dimensionality Reduction:

Use techniques like Principal Component Analysis (PCA) to reduce the dimensionality of the feature space while preserving important information.

Feature Selection:

Apply feature selection techniques to identify the most relevant features and reduce noise. Common methods include Recursive Feature Elimination (RFE) and feature importance from tree-based models.

Interaction Features:

Create interaction features by combining two or more existing features to capture relationships that the model may not discover on its own.

Time-Series Features:

Calculate time-series-specific features such as autocorrelation, cross-correlation, or seasonality indices to capture temporal patterns.

Missing Data Handling:

Address missing data by imputing values using methods like mean, median, or more advanced techniques like k-nearest neighbors or regression imputation.

Feature Engineering Feedback Loop:

Continuously evaluate and refine your feature engineering process based on model performance and domain knowledge.

Feature Importance:

Use feature importance scores from machine learning models (e.g., Random Forest, XGBoost) to identify which features have the most impact on predictions.

Regularization:

Apply L1 or L2 regularization to reduce overfitting by penalizing the importance of certain features.

Remember that feature engineering is an iterative process, and it's essential to work closely with domain experts to create features that reflect the underlying factors influencing production demand. Experiment with different feature combinations and engineering techniques to find the best set of features for your specific demand prediction problem.

Model Training

The process of training an ML model involves providing an ML algorithm (that is, the learning algorithm) with training data to learn from. The term ML model refers to the model artifact that is created by the training process.

The training data must contain the correct answer, which is known as a target or target attribute. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict), and it outputs an ML model that captures these patterns.

Linear Regression

Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

This form of analysis estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values. There are simple linear regression calculators that use a “least squares” method to discover the best-fit line for a set of paired data. You then estimate the value of X (dependent variable) from Y (independent variable).

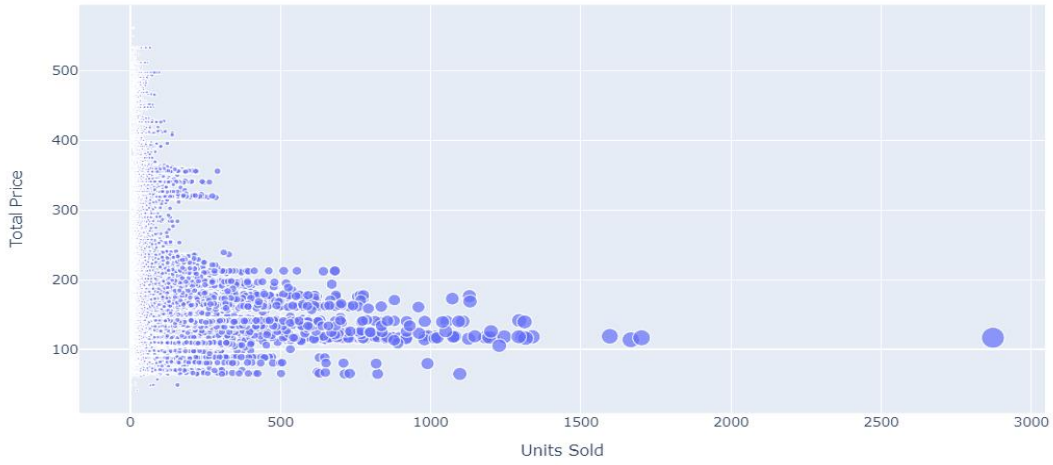
```
from sklearn.linear_model import LinearRegression
import sklearn.metrics as sm
regr=LinearRegression()
regr.fit(xtrain, ytrain)
print("r2 score=",sm.r2_score(ytest,y_pred))
```

```
In [40]: from sklearn.linear_model import LinearRegression
         import sklearn.metrics as sm
         regr=LinearRegression()
         regr.fit(xtrain, ytrain)
         print("r2 score=",sm.r2_score(ytest,y_pred))

         r2 score= 0.14516652987722223
```

Let us now analyze the relationship between the price and the demand for the product. Here I will use a scatter plot to see how the demand for the product varies with the price change:

```
fig = px.scatter(data, x="Units Sold", y="Total Price",size='Units Sold')

fig.show()
```

```
In [14]: fig = px.scatter(data, x="Units Sold", y="Total Price",size='Units Sold')
         fig.show()
```



DECISION TREE

A decision tree is a flowchart-like tree structure where each internal node denotes the feature, branches denote the rules and the leaf nodes denote the result of the algorithm. It is a versatile supervised machine-learning algorithm, which is used for both classification and regression problems.

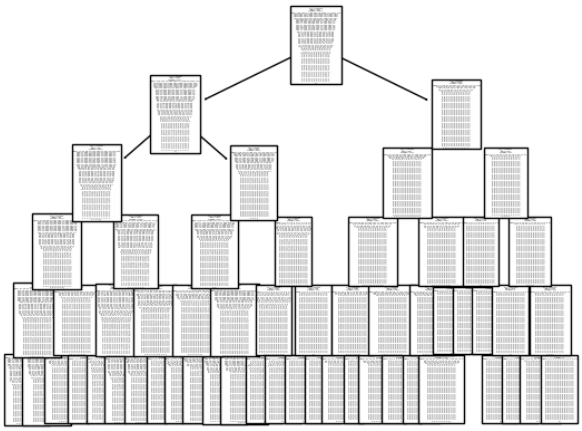
DECISION TREE CLASSIFIER

use the decision tree classifier algorithm to train our model

```
from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

clf=DecisionTreeClassifier(max_depth=5)

clf1=clf.fit(xtrain,ytrain)

y_predict=clf1.predict(xtest)

fig=plt.figure()

tree.plot_tree(clf1)

plt.show()
```

fig.savefig("decision_tree.png")

```
In [45]: from sklearn.tree import DecisionTreeClassifier
         from sklearn import tree
         clf=DecisionTreeClassifier(max_depth=5)
         clf1=clf.fit(xtrain,ytrain)
         y_predict=clf1.predict(xtest)
         fig=plt.figure()
         tree.plot_tree(clf1)
         plt.show()
         fig.savefig("decision_tree.png")
```



finding the accuracy of the model

from sklearn.metrics import accuracy_score

print("Accuracy:")

accuracy=accuracy_score(ytest,y_predict)

print(accuracy*100,"%")

```
In [46]: from sklearn.metrics import accuracy_score
         print("Accuracy:")
         accuracy=accuracy_score(ytest,y_predict)
         print(accuracy*100,"%")

         Accuracy:
         2.173382173382173 %
```

Evaluation

use the decision tree regression algorithm to train our model

from sklearn.tree import DecisionTreeRegressor

model = DecisionTreeRegressor()

```
In [27]: from sklearn.tree import DecisionTreeRegressor
         model = DecisionTreeRegressor()
         model.fit(xtrain, ytrain)

Out[27]: DecisionTreeRegressor()
```

Now let's input the features **(Total Price, Base Price)** into the model and predict how much quantity can be demanded based on those values:

model.fit(xtrain, ytrain)

y_pred = model.predict([[190.2375,234.4125]])

print("units sold(predicted): % d\n"% y_pred)

In [28]:
```
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()
model.fit(xtrain, ytrain)
y_pred = model.predict([[190.2375,234.4125]])
print("units sold(predicted): % d\n"% y_pred)
```

```
units sold(predicted):  41
```