MODEL-CENTRIC ML BIG PICTURE

1) Data Split

In model-centric machine learning, the first step is **splitting the data** into training and testing sets to ensure that we are training the model on one portion of the data and testing it on a separate, unseen portion, to simulate future data. Here's the breakdown:

- **Data Split Ratios**: You typically split the dataset in a ratio like **75-25 or 80-20**, where 75% (or 80%) is the training set, and 25% (or 20%) is the testing set.
    - **Training Set**: This is used for **building the model**. However, we go a step further by splitting the training set again into:
        - **Train (Analysis) Set**: Used to train the algorithm.
        - **Test (Assessment) Set**: This helps in **cross-validation**, where you assess how well the algorithm is generalizing before using the test set. Cross-validation techniques like K-fold or stratified sampling are used here.
- **Testing Set**: This set simulates **future data** and provides a final evaluation of the model performance after all model building steps.

---

2) Feature Engineering

Feature engineering is crucial for improving model performance. We aim to apply the **Big-5** techniques to ensure our machine learning model uses well-prepared and meaningful features. The **Big-5** are:

a) Zero-Variance Feature Removal:

- If a feature has **zero variance** (i.e., the same value for all observations), it provides no useful information for the model, and we should remove it.

b) Imputation of Missing Values:

- **Missing data** is a common issue in real-world datasets. We either:
    - **Remove** records with missing data (if minimal).

- **Impute** them using techniques like the mean, median, mode, or advanced methods like k-NN imputation.

c) Normalization (Standardization):

- Many algorithms perform better when features are normalized, particularly algorithms that are sensitive to scales like SVM, k-NN, and neural networks.
- We transform the data into a **standard normal distribution** with a mean of 0 and a standard deviation of 1.

d) Transformation:

- If the input or output data has significant **skewness**, it can affect model performance. Transformation techniques like **logarithmic**, **Box-Cox**, or **exponential** transformations can make the distribution more normal, improving algorithm performance.

e) Encoding:

- **Categorical features** need to be encoded as numeric values. There are **7 methods** of encoding:
    1. **One-hot encoding**: Converts categories into binary columns.
    2. **Label encoding**: Assigns each category a unique number.
    3. **Frequency encoding**: Replaces each category by its frequency.
    4. **Target encoding**: Uses the mean of the target variable per category.
    5. **Ordinal encoding**: Assumes an inherent order between categories.
    6. **Binary encoding**: Combines the benefits of label and one-hot encoding for high cardinality categories.
    7. **Hash encoding**: Encodes features into fixed-size hash codes.

Beyond the Big-5, you also have advanced techniques:

a) Feature Signal Extraction:

- **Dimension Reduction**: Reducing the number of features while retaining as much information as possible. Linear methods include:
    - **PCA** (Principal Component Analysis),
    - **ICA** (Independent Component Analysis),

    - **PLS** (Partial Least Squares).
- Non-linear methods include:
    - **UMAP** (Uniform Manifold Approximation and Projection),
    - **t-SNE** (t-distributed Stochastic Neighbor Embedding).

b) Feature Selection:

- This technique identifies the most **relevant features** for the model. A popular method is **Recursive Feature Elimination (RFE)**, which recursively removes less useful features until the best subset is selected.

c) Feature Development:

- Sometimes, new features need to be developed based on **domain knowledge and experience**. This involves methods like **alpha search**, which focuses on discovering interactions or new representations of the data that improve model accuracy.

---

3) Model Selection (Fitting)

Once the data is ready, we move to **model selection**. Different models are suited for different tasks:

- **Neural Networks** are generally best for **non-tabular data** (e.g., images, text, audio) because they excel in capturing complex patterns and non-linear relationships.
- For **tabular data**, algorithms like **Random Forest, Gradient Boosting Machines (GBM), Support Vector Machines (SVM)**, and **XGBoost** tend to perform well. The choice of algorithm depends on the data structure and the problem.

---

4) Hyperparameters Tuning

After selecting a model, the next step is tuning its **hyperparameters**. Hyperparameters are the settings of the model that control the training process. Fine-tuning these parameters ensures optimal performance:

- **Grid Search** or **Random Search** are common techniques for hyperparameter tuning.
- We use **cross-validation** (splitting the data again within the training set) to ensure that the model is not **overfitting** to the training data and generalizes well to new data.

5) Model Assessment

Once the model has been trained and tuned, we need to assess its performance using the testing set. Key evaluation metrics for **regression problems** include:

- **MAE** (Mean Absolute Error): Average of absolute errors between predicted and actual values.
- **MAPE** (Mean Absolute Percentage Error): Average of the percentage errors between predicted and actual values.
- **RMSE** (Root Mean Squared Error): The square root of the average of squared differences between the predicted and actual values.

For **classification problems**, other metrics like **accuracy**, **precision**, **recall**, and **F1-score** would be used.