Feature Engineering in ML explained in simple terms and how to implement it (with code)

✅Feature Engineering in ML explained in simple terms and how to implement it (with code).
A quick thread 🧵👇
#Python #DataScience #MachineLearning #DataScientist #Programming #Coding #hubofml #deeplearning

1/ Why Feature Engineering?

Improved Model Performance: By engineering informative and representative features, we can provide the model with more relevant information, enabling it to learn more effectively.

2/ Noise Reduction: Raw data often contains noise, irrelevant information, or redundant features that can hinder model performance. Feature engineering helps in selecting or creating features that are more informative and discriminative, filtering out unnecessary noise.

3/ Handling Non-linearity: Many real-world problems have non-linear relationships b/w input features and the target variable. Feature engineering allows us to create non-linear transformations,interaction terms, or feature crosses that can help capture these complex relationships.

4/ Dealing with Missing Data: Real-world datasets often contain missing values. Feature engineering techniques like imputation can help fill in missing data points, enabling the model to utilize the available information more effectively.

5/ Handling Categorical Data: ML models generally require numerical inputs, but real-world datasets contain categorical variables. FE techniques like one-hot encoding or ordinal encoding transform categorical variables into numerical representations that models can process.

6/ Dimensionality Reduction: FE techniques like PCA, SVD, or feature extraction help in reducing dimensionality of the feature space. This can be beneficial for several reasons,such as reducing computational complexity, avoiding overfitting, and visualizing high-dimensional data.

7/ Interpretability: Well-engineered features can provide better interpretability of the model's predictions. By designing features that are meaningful and aligned with the problem domain, we can gain insights into the factors influencing the model's decisions.

8/ Feature engineering is the process of creating new features or transforming existing features from raw data to improve the performance of machine learning

models. It involves selecting, creating, and manipulating features to make them more informative, discriminative.

9/ How to do Feature Engineering?

Feature Selection: Selecting the most relevant features can help reduce noise and improve model performance. Techniques like correlation analysis, statistical tests, and domain knowledge can be used for feature selection.
Implementation -

10/ Feature Extraction: Extracting new features from existing data can provide additional information to the model. Methods such as Principal Component Analysis , Singular Value Decomposition (SVD), and t-SNE (t-distributed Stochastic Neighbor Embedding) can be employed.
Code-

11/ Handling Missing Data: Missing data can be a common problem in real-world datasets. It's important to handle missing values appropriately, either by imputing them (using techniques like mean, median, or regression imputation) or marking them as a separate category.
Code -

12/ Encoding Categorical Variables: Many machine learning algorithms require numerical inputs, so categorical variables need to be encoded. Common methods include one-hot encoding, ordinal encoding, and binary encoding.
Code -

13/ Scaling and Normalization: It's necessary to scale numerical features to a standard range to avoid biasing the model towards features with larger magnitudes. Standardization (mean 0, standard deviation 1) and min-max scaling (between 0 and 1) can be used for feature scaling.

14/ Binning and Discretization: Continuous numerical features can sometimes be discretized into bins or categories, which can capture non-linear relationships. This can be done using techniques like equal-width/binning, equal-frequency/binning, or using domain-specific knowledge

15/ Feature Engineering from Time Series: Time series data may require special handling, such as creating lag features (using past values as new features) or rolling window statistics (e.g., mean, max, min over a specific time window).
Implementation -

16/ Feature Crosses: Creating interaction terms or combinations of features (feature crosses) can help capture non-linear relationships between variables. This can be useful in tree-based models or models that can handle feature interactions.
Implementation -

17/ Feature Importance: After training a model, you can assess importance of different features in predicting target variable. Permutation importance, feature importance from tree-based models,or feature importance from linear models can help identify most influential features.

19/ Some feature Engineering tips -
Avoid data leakage: Be cautious of data leakage, which occurs when info frm target variable is inadvertently present in features. Ensure that feature engineering is done

based on information that would be available during model deployment.

20/ Split your data carefully: Split your data into training and test sets before performing any feature engineering. This helps prevent information leakage and allows you to evaluate the effectiveness of your engineered features properly.

21/ Consider feature selection: Select the most relevant features to reduce noise and improve model performance. Techniques like correlation analysis, statistical tests, and domain knowledge can be used for feature selection.

22/ Scale numerical features: Scale numerical features to a standard range to avoid biasing the model towards features with larger magnitudes. Techniques like standardization (mean 0, standard deviation 1) or min-max scaling (between 0 and 1) can be employed.

23/ Continuously iterate and refine: Feature engineering is an iterative process. Continuously evaluate and refine your features based on the performance of your models. Experiment with different techniques and evaluate the impact on model performance.

24/ More - https://t.co/Ij0wULOXQH
