ML

Prepare the data for ML algorithms:-

Data Preparation involves:-

1. Data Cleaning
2. Data Transformation
3. Data reduction
4. Feature engineering
5. Data splitting

1. Data Cleaning:-

This involves

(i) Handling missing values

(ii) Handling Outliers

Def: * It is the process of identifying & correcting errors, inconsistencies & missing values in a dataset to improve its quality & reliability for analysis & modeling.

Why data cleaning is important?

1. Accuracy:- Accurate model training & reliable predictions.
2. Quality:- For better decision-making
3. Normalization:- Helps in standardizing & preparing dataset for analysis.
4. Feature engineering:- Effective feature extraction & selection.
5. Overall efficiency of model is increased.

(i) Handling missing values:-

Missing values:- Absence of information or data points for certain observations or attributes in a dataset.

How to handle them?

1. Deletion of entire rows involving missing values. But it leads to loss of valuable data.
2. Mean/Median/Mode Imputation:- This may distort original distribution.
3. Forward fill / Backward fill:- Fill missing values with most recent non-missing value (forward fill) or the next non-missing value along the column (backward fill)

④ Prediction Model: Using ML algorithms to predict missing values based on other features in the dataset. This requires more computational resources.

⑤ K-nearest Neighbors (KNN):- Values of nearest neighbors in feature space

Program.

combined f^n in scikit-learn

imputer.fit_transform(df) → Calculates mean for each column & replaces missing values with those mean

The result is numpy array. So we wrap it with pd.DataFrame(...) & give it same column name

```
array([
    [10.0, 50],
    [20.0, 16.25],
       :
])
```

} Numpy array

Triveni TURBINES

(ii) Handling outliers:- Outliers are data points that significantly deviate from other observations in a dataset. Reason:- Measurement errors, data entry mistakes or genuine extreme values in data

How to identify outliers?

(1) Visual methods → Plotting or z score/IQR (Interquartile Range) & Tukey's method.

Methods to handle outlier:

(1) Removing outliers:- z scores, IQR
Careful considerations are needed as genuine data might get lost.

(2) Transforming data. log transformations, square root transformation.

(3) Capping or Winsorizing. When data points are valid & needs to be controlled in the analysis.
Capping:- Setting a threshold beyond which values are capped.
Winsorizing:- Replaces extreme values with a specified percentile.

(4) Binning:- Grouping outliers in to a separate category.

(5) Advanced Models: Random Forest or SVM are less sensitive to outliers.

Triveni TURBINES

Triveni TURBINES

Program.

① data is a dictionary data with key "Values" & its value as a list of numbers.

② 150 & 200 might be outliers

③ pd.DataFrame(data) → data dictionary to Pandas dataframe called df which is a table-like structure

④ df['Z_score'] = (df['Values'] - df['Values'].mean()) / df['Values'].std(ddof=0)

This calculates Z-score for each value in Values column & adds it as a new column Z-Score

Triveni TURBINES

Z-Score tells us how far each data point is from the mean, in terms of standard deviation

ddof = 0 → population standard deviation.
(sample standard)

Standard deviation (SD) is a measure of how spread out numbers in a dataset are from mean (average). If SD is small, most numbers are close to mean or if it is large, n.o.s are spread out.

PSD: You have all data from a population

⑤ Threshold is set what is considered an outlier.

Any threshold with Z-score > 1 or < -1 is treated as outlier.

Triveni TURBINES

(6) Filtering dataframe by a condition

df_cleaned = df[np.abs(df['Z_score']) <= threshold]

This creates new dataframe df_cleaned that only includes rows where the absolute z-score is ≤ 1.

abs → absolute value

Conditional filtering :- or boolean masking

df[...] → Selects rows from df based on a condition inside the brackets.

np.abs(df['Z_Score']) <= threshold
↳ This creates a boolean Mask
[True, True, True, False, T, T, F, T]
Keep only rows that is true.

Alternate

df_cleaned = df.loc[np.abs(df['Z_Score']) <= threshold, ['Values']]

Triveni TURBINES

Z-Score

Values = [10, 20, 30, 150, 25, 35, 200, 40]

Step 1:

$$\text{Mean} = \frac{10+20+30+ \dots +40}{8} = \frac{510}{8} = 63.75$$

Step 2:

Value	Difference from Mean	Squared difference
10	10 − 63.75 = −53.75	$(-53.75)^2 = 2889.06$
20		
30		
⋮		
150	150 − 63.75 = 86.25	7439.06
⋮		
200	200 − 63.75 = 136.25	18563.06
40	−23.75	564.06

Step 3: Calculate variance (Average of squared differences)

Variance = 4542.06.

Step 4: Standard deviation = $\sqrt{4542.06} \approx 67.35$

This is population standard deviation

Mean → 63.75
SD → 67.35

Most values are within 1 SD (±67.35) from Mean.

Now z-score:

$$z = \frac{x - \text{Mean}}{\text{Standard deviation}}$$

$$z = \frac{10 - 63.75}{67.35} \approx -0.8$$

① Predicting whether an email is spam (positive class) or not spam (negative class). (Binary classification problem)

* After applying a classification algorithm to a set of emails, we can construct a confusion matrix to evaluate its performance

TN → Emails correctly predicted as not spam
FP → " incorrectly " " spam (actually not spam)
FN → Emails incorrectly predicted as not spam (actual spam)
TP → Emails correctly predicted as spam

Suppose we have 100 emails, 30 → spam 70 → not spam
After applying a classification algorithm, the confusion Matrix:

Actual / Predicted	Predicted Not Spam	Predicted Spam
Actually not spam	65	5
Actual spam	10	20

Triveni TURBINES

Data Transformation: Modifying original data to make it suitable for analysis or modeling

Why data transformation is necessary?

① Feature scaling: Adjusting range of features in the data.

Ex:- Different features in the dataset might have different units & scales.
Age → 0 to 100
salary → thousands to crores.
KNN & SVM are biased towards features with larger scales.
Transformations like Min-Max scaling or Standardization help normalize data ensuring that each feature contributes equally to model's predictions.

Triveni TURBINES

② Handling skewed data:- Logarithmic, square root or Box-Cox can reduce Skewness.

③ Encoding categorical variables: ML models generally work with numerical data. Categorical data such as gender (Male/female) or state names need to be converted to numerical formats using ~~hot~~ one-hot encoding or label encoding

④ Feature extraction: Helps in extracting more Meaningful features.

~~⑥~~ ⑤ Performance & accuracy of model is increased.

Common methods of data transformation

① Normalization: Scaling ~~values~~ numerical features values to standard range (0 & 1)

Ex: House prices brought to 0 & 1
(100,000 to 2,00,000)

This helps in fair comparison with other features

2. Standardization:- It centers data around a mean of 0 & scales it to have a standard deviation of 1.

* Beneficial for algorithms that assume normally distributed data.

3. Log Transformation:- ~~log~~ It is applied to skewed data, like converting income values to their logarithmic ~~fa~~ form to handle extreme values.

4. Encoding Categorical Variables:-

One-hot encoding:- Creates new binary column for each category level

label encoding: Assigns a unique integer based on alphabetical ordering of categories.

Triveni TURBINES

Gender
Male
Female
Male
Male
Female

⟹

Male	Female
1	0
0	1
1	0
1	0
0	1

Categorical feature with 2 categories

Categorical feature converted to 2 binary features

Ex: Dataset

Student ID	Math Score	English Score
1	85	40
2	70	30
3	90	45
4	65	25
5	80	35

Normalizing exam score bet^n 0 & 1

Data transformation steps:-

① Feature scaling:-

$$\text{Normalized Value} = \frac{\text{Value} - \text{Min Value}}{(\text{Max Value}) - (\text{Min Value})}$$

② Maths.

$(85-65)/(90-65) = 0.75$

③ English

$(\cancel{4}\ 40-\cancel{85})/(45-25) = 0.75$

Program:

① StandardScaler:- To standardize Age & Income (Numerical Pipeline)

② OneHotEncoder:- To encode 'Gender' & Region (Categorical pipeline)

③ ColumnTransformer (Full pipeline):- It combines numerical & categorical pipelines to apply transformations to respective columns. It allows applying different preprocessing to different columns.

④ Pipeline: It allows chaining transformation in sequence

Triveni TURBINES

```
num_pipeline = Pipeline([

    'scaler', StandardScaler())
])
```

scaler → name for step

StandardScaler → Normalizes values

It contains single step

Same for OneHotEncoder()

Combine pipelines with columnTransformer → applies correct pipeline to right column

```
full_pipeline = ColumnTransformer([

    ('num', num_pipeline, ['Age', 'Income']),
    ('cat', cat_pipeline, ['Gender', 'Region'])
])

transformed_data = full_pipeline.fit_transform(df)
```

Triveni TURBINES

Data Reduction:- Process of diminishing amount of data that needs to be processed & analysed without sacrificing valuable information.

Why data reduction is important ??

1. Improves efficiency:- Computational resources requirement reduces significantly. Quicker execution of data analysis tasks
2. Reduces storage requirements:- Important for businesses
3. Enhances model performance: Helps in removing irrelevant or redundant features which improves model's accuracy & performance. Principal Component Analysis (PCA) & feature selection are commonly used.
4. Mitigates overfitting:- Overfitting occurs when a model learns not only valid patterns but also noise in training data.

By reducing no. of features, risk of overfitting is reduced; Thereby it can make good predictions on new unseen data

5. Simplifies data visualization:- Visualizing few variables can help in drawing more meaningful conclusions.

6. Improves data quality:-

7. Cost effective data management:- Storage & computational cost reduced.

Common methods of data reduction:-

1. Feature selection:- Discarding irrelevant features & choosing a subset of relevant features
2. Feature extraction:- Transforms original features into lower-dimensional space to capture essential information.
3. Instance selection:- Choosing a subset of representative instances from the dataset while maintaining overall data characteristics. This speeds up training
4. Dimentionality reduction:- Reduce the number of input variables/features in a dataset.

Example:-

Dataset → name, age, DOB, Score, study hrs, extracurricular activities & academic Performance

Goal is to predict student grades based on these features.

Most influential features:- exam scores, study hours & extracurricular activities

Key features: exam scores, study hours

Triveni TURBINES

Code

1 Imports

```
sklearn.feature_selection import selectKBest,
                                 f_classif
```

↳ SelectKBest, f_classif: Used to select most relevant features (column) based on statistical tests (ANOVA F-test)

```
from sklearn.decomposition import PCA
```

↳ Principal Component analysis: Used to reduce no. of features while retaining most of information

```
from sklearn.preprocessing import StandardScaler
```

↳ Scales features to have 0 mean & unit variance

Triveni TURBINES

2 `np.random.seed(42)`

↳ Ensuring reproducibility

3 data: Synthetic data generation (fake but realistic)

`np.random.normal(5, 2, 100)`

↳ np.random.normal(mean, std, size)

↳ This means,

average study hours = 5 hrs.

Most values will be bet? 3 to 7 hours (±2 std devs)

100 such values are generated

`np.random.rand(size)`

↳ This generates uniformly distributed random values bet? 0 & 1

↳ np.random.rand(100)*10

This generates 100 values bet? 0 & 1 then multiplies by 10.

Triveni
TURBINES

(4) Creating a binary target variable

```
data['Pass'] = ((data['TestScores'] + data['Assignments']
                + data['ProjectScore']) / 3 > 50)
                .astype(int)
```

↳ We are computing average score from 3 performance metrics.

↳ If the average is > 50, student passes (1), else fails (0)

↳ astype(int) → 1s & 0s. (binary target)

(5)

```
scaler = StandardScaler()
features_scaled = scaler.fit_transform
        (data.drop('Pass', axis=1))
```

↳ StandardScaler() Standardizes each feature to a Mean of 0 & std. dev of 1

Triveni
TURBINES

We drop Pass column since its target not a feature

fit_transform() computes scaling parameters & applies them.

(6) Feature selection (Top 3 best features)

```
selector = SelectKBest(score_func = f_classif, k=3)
selected_features = selector.fit_transform(features_scaled,
                    data['Pass'])
```

↳ SelectKBest selects top 3 features most correlated with Pass target using ANOVA F-test.

↳ fit_transform() finds the best features & returns the scaled values for them.

Triveni TURBINES

(7) feature_names = data.columns[:-1]
selected_features_names = feature_names
[selector.get_support()]

↳ feature_names → grabs all columns names except last one (Pass)

↳ selector.get_support() → Returns boolean mask indicating which features were selected

(8) Dimensionality reduction with PCA.

```
pca = PCA(n_components=2)
features_pca = pca.fit_transform(features_scaled)
```

↳ Creates PCA object to reduce the data to 2 principal components

↳ fit_transform() applies PCA on scaled data to return transformed data

Triveni TURBINES

(9) ① Original data shape → data.drop('Pass')

② Data shape after feature selection → selected_features

③ print actual feature names selected by SelectKBest → selected_feature_names

④ Data shape after PCA → features_pca

Triveni TURBINES

① Diff Selected features & features_pca

② Virtual environment

③ ANOVA F-Test

① features_pca → creates brand new features by combining existing features in a smart way

PC1 = 0.4 * TestScores + 0.3 * Attendance − 0.2 * Participation + ..

They don't have names like original columns, because they are combinations of all of them

② selected_features → Keeps actual original features
Selects only top k features that are related to Pass

Triveni TURBINES

ANOVA F-test → Analysis of Variance
→ Does a feature significantly affect the target variable?

f_classif → ~~pastivt~~ checks which features help predict whether a student passes or not

When Use?
When target is categorical
Features are numeric

F = Variance bet^n groups / Variance within groups

Feature Engineering :- Here new features are created from existing data, transforming data into more useful formats or enhancing quality of data to improve efficiency & accuracy of models.

Key Components

① Feature Creation :- To provide additional insights to the models.
Creation :- Combining features, deriving new metrics from existing data or aggregating data over time/space

Ex :- A dataset containing student attendance & grades, creating a new feature represents the average grade over past 3 tests might predict future performance than individual test scores.

② Feature Transformation : Normalization, scaling
Ex : In a dataset, transforming 'StudyHours' feature from raw hours to categories such as Low, Medium & High based on thresholds (eg: 0-3, 4-6, 7+ hours) can sometimes provide clearer signals for predicting student performance

Why Feature engineering is important in ML?

① Improves Model Performance :- Better Pattern representation
② Reduces model complexity by capturing underlying patterns in data.
③ Enhances data interpretability
④ Adaptability across various models → Without changing underlying algorithms

Triveni TURBINES

Program

1. Import
2. Data – 100 rows
3. Feature transformation – Normalizing values using binning

```
data['NormalizedStudyHours'] = pd.cut
    (data['StudyHours'], bins = 3,
                         labels = [1, 2, 3])
```

pd.cut () divides continuous data into discrete intervals

4.
```
data['NormalizedSleepHours'] =
        pd.cut(data['SleepHours'],
    bins=[0, 6, 8, np.inf], labels=[1, 2, 3])
```

5.
```
data['NormalizedExerciseHours'] =
    pd.cut(data['ExerciseHours'], bins=3,
           labels = [1, 2, 3])
```

Triveni TURBINES

6. Feature creation – Health index

```
data['HealthIndex'] = (
        data['NormalizedSleepHours'].astype(int) +
        data['NormalizedExerciseHours'].astype(int)
        ) / 2
```

* .astype converts labels (1, 2, 3) from category to integer

* This index reflects overall lifestyle health.

7. Feature creation – Adjusted GPA :-

```
data['AdjustedGPA'] = data['GPA'] + (
    (data['NormalizedStudyHours'].astype(int) - 2)
        * 0.1)
```

* Subtracting 2 :- This sets the medium level (2) as the baseline (no adjustment).

Triveni
TURBINES

GPA	Normalized Study Hours	Converted to int	Value −2	Bonus	Adjusted GPA
2.8	1	1	−1	−0.1	2.7
3.2	2	2	0	0	3.2
3.5	3	3	+1	+0.1	3.6

This column makes hidden patterns more visible to the model.

GPA = 3.5
Normalized Study Hours = 3 (high)

Then:

$$\text{Adjustment} = (3-2) \times 0.1 = 0.1$$
$$\text{Adjusted GPA} = 3.5 + 0.1 = 3.6$$

* Simple behavioral modeling.
* How much a student studies has a small positive or negative effect on their GPA.

Triveni TURBINES

Data Spitting:- Dividing data into separate datasets. Goal → Model trained on one set of data can generalize well on new, unseen data.

Types of data splits:-

① Training data ② Validation data ③ Test data.

Training data:- Largest portion of dataset. It is used to train the model. Identify patterns & make decisions based on this data. 70-80%.

Validation data:- Used to tune the model's hyperparameters & make decisions about which models or configurations to use. 10-15%.

Test data:- Used to evaluate final model's performance after it has been trained & validated. Test set should be completely independent dataset that the model hasn't seen during training/validation. 10-15%.

Triveni TURBINES

Methods of data splitting:

① Random splitting:- Randomly assigning data points to training, validation & test sets.

Assumption:- All data points are independent & identically distributed.

② Stratified splitting:- Data points are unevenly distributed for example in classification problems with imbalanced classes.

This splitting ensures that each class is proportionately represented in training, validation & test sets.

Model developed will be fair & has learned adequately from all classes.

Triveni TURBINES

③ Time-based splitting:- Data is split based on time. Temporal patterns & dependencies are important.
Ex. Model may be trained on data from past year, validated on the following month & tested on the month after that.

Why data splitting is important?

① Avoiding overfitting:- Separate datasets
② Model validation:- Avoids biased assessments. Identifies best model settings.
③ Assessing model performance
④ Improving model robustness:- Handles variations in data & accurately predict outcomes across a range of scenarios.
⑤ Effective tuning & comparison

Triveni TURBINES

Select & train a model:

* The objective is to ensure that model not only performs well on training data but also generalizes effectively to unseen data.

* Prepared data is used to create a model capable of making predictions based on new i/ps.

Process of selecting & training a ML model

① Model Selection:-
* Choice depends on nature of problem.
* Whether problem type ~~depends on~~ involves regression, classification, clustering or other tasks.
* Predicting student performance (score prediction) → regression models are used.

Triveni TURBINES

Ex:- Start with simpler models like linear regression. If relationship bet^n features & target ~~are~~ is non-linear, then random forest regression is used due to its ability to handle complex data structures.

② Model training:-

* Selected model is exposed to the prepared dataset to learn patterns & relationships bet^n input features & target variables.
* Here internal parameters are adjusted by the model on the training data which minimizes prediction error & improve its performance.

Ex:- The random forest model is trained using features such as study hours, attendance records & historical grades. This model doesn't require setting

Triveni TURBINES

Many hyperparameters but involves using decisions about no. of trees & depth, which we initially set to defaults for a baseline model.

③ Model Evaluation:-

* Validation set is used to evaluate model. Its ability to generalise on new data (Unseen) is evaluated.
* Evaluate initial random forest model by calculating RMSE on validation set.
* If performance is unsatisfactory, need of tuning hyperparameters arises.

④ Hyperparameter tuning:-

* These are parameters which are set before learning process begins.
* They control learning process & model behavior. but are not learned from data.
* Ex:- learning rate, regularization strength, no. of hidden layers in neural n/w & kernel type

Triveni TURBINES

Ex:- Adjusting hyperparameters like no. of trees or tree depth in a random forest model through grid search can help minimize RMSE on validation set.

(5) Cross-Validation :- Model's stability across various data subsets.

* Repeatedly splitting data into training & validation sets, training on each subset & averaging results.
* Ex:- 10-fold cross-validation on random forest
  ↳ Dividing training data into 10 subsets, using each as validation set once & training on other remaining 9 subsets.
  ↳ Average RMSE.

Triveni TURBINES

(6) Final model training:- After identifying best Model & hyperparameters, final model is trained on the entire training dataset.

Ex:- The random forest model with fine-tuned hyperparameters from cross validation is trained on complete student dataset to maximize its predictive capabilities.

(7) Model Testing:- Final test for random forest model involves predicting exam scores for new students based on their study habits & past performance. Model's predictions are compared against actual scores to calculate final RMSE.

```
study_hours = np.array([1, 2, 3, 4, 5]).reshape(-1,1)
```

array ↳ no. of hrs students studied

* .reshape(-1, 1) → reshapes it to 2D array because sklearn expects the i/p features in 2D (rows = samples, columns = features)

* scores → 1D numpy array → test scores for each set of study hrs.

$$\boxed{\text{Model = LinearRegression()}}$$

↳ Creates an instance of LinearRegression model

$$\boxed{\text{Model.fit(study\_hours, scores)}}$$

↳ trains model. It finds best-fit line to map study hrs to scores.