EDA and project Explanations :

EDA and project Explanations :

1. **False Positive (FP)**:
    - Definition: A false positive occurs when the model incorrectly predicts the positive class (1) for an instance that actually belongs to the negative class (0).
    - Example: In a medical diagnosis scenario, a false positive would be when the model predicts a patient has a disease (positive class) when they do not actually have it (negative class).
2. **False Negative (FN)**:
    - Definition: A false negative occurs when the model incorrectly predicts the negative class (0) for an instance that actually belongs to the positive class (1).

- Example: Continuing with the medical diagnosis example, a false negative would be when the model predicts a patient does not have a disease (negative class) when they actually do have it (positive class).

1. **Handling many Null values in almost all columns**:
    - This suggests that the dataset contains a substantial amount of missing data, which needs to be addressed before building a predictive model. Various strategies such as imputation, deletion, or advanced techniques like multiple imputation may need to be employed to handle these missing values effectively.
2. **No low-latency requirement**:
    - Low-latency requirements typically mean that predictions need to be made quickly, often in real-time or near-real-time. In this case, since there's no such requirement, the focus can be on building accurate models even if they take longer to train or make predictions.
3. **Interpretability is not important**:
    - This implies that the primary objective is predictive accuracy rather than understanding the underlying factors driving the predictions. Models that are highly accurate but complex, such as deep neural networks or ensemble methods, may be preferred over simpler, more interpretable models like logistic regression or decision trees.
4. **Misclassification leads to unnecessary repair costs**:
    - This indicates that the cost of misclassifying instances, particularly false positives and false negatives, is associated with repair costs. For example, in a manufacturing setting, misclassifying a defective product as non-defective (false negative) could lead to costly repairs or replacements down the line. Similarly, incorrectly classifying a non-defective product as defective (false positive) could result in unnecessary repair costs.

KNN imputer

The **KNNImputer** is a method used for imputing missing values in a dataset based on the k-nearest neighbors approach. It replaces missing values in a feature with the mean value of the feature's k-nearest neighbors. Here's how it works:

1. For each sample with missing values, the algorithm finds its k-nearest neighbors based on other samples with non-missing values.
2. It calculates the mean value of the feature among these k-nearest neighbors.
3. This mean value is then used to replace the missing value in the original sample.

1. **StandardScaler**:
    - Standardizes features by removing the mean and scaling to unit variance.
    - Sensitive to outliers because it computes the mean and standard deviation, which can be influenced by extreme values.
    - Works well when the data is normally distributed and does not contain many outliers.
2. **RobustScaler**:
    - Scales features using statistics that are robust to outliers.
    - It uses the median and interquartile range (IQR) instead of the mean and standard deviation.
    - More suitable for data with outliers or skewed distributions because it's less affected by extreme values.
    - It's not sensitive to the presence of outliers as it focuses on the median and qua

1. **MinMaxScaler**:
    - Scales features to a specified range, typically between 0 and 1.
    - It's sensitive to outliers because the scaling is based on the minimum and maximum values of each feature.
    - Preserves the shape of the original distribution, but it may not handle outliers well.
2. **RobustScaler**:
    - Scales features using statistics that are robust to outliers, such as the median and interquartile range (IQR).
    - It's less affected by outliers compared to MinMaxScaler because it uses more robust statistics.
    - It's suitable for data with outliers or skewed distributions because it adjusts the scaling based on the median and quartiles.

1. **SMOTE (Synthetic Minority Over-sampling Technique):** SMOTE works by creating synthetic samples from the minority class (the class with fewer instances) rather than duplicating them. It randomly selects a minority class instance and finds its k nearest minority class neighbors. Then, it randomly selects one of these neighbors and creates a synthetic instance along the line segment joining the two points in the feature space. This process helps to balance the class distribution by increasing the number of minority class instances.
2. **Tomek links:** Tomek links are pairs of instances from different classes that are very close to each other in the feature space. By removing such instances, Tomek links can help clarify the decision boundary between classes. In the context of SMOTE, applying Tomek links after generating synthetic samples helps to improve the quality of the synthetic data by removing noisy or ambiguous instances.

SMOTE+TOMEK: SMOTE+TOMEK is a combination of SMOTE oversampling and Tomek links undersampling. It first applies SMOTE to oversample the minority class, and then applies Tomek links to remove potentially noisy or ambiguous instances from both the minority and majority classes. This combined approach aims to create a more balanced and clearer representation of the dataset

Steps you need to follow :

- Reading the data in python
- Defining the problem statement
- Identifying the Target variable
- Looking at the distribution of Target variable
- Basic Data exploration
- Rejecting useless columns
- Visual Exploratory Data Analysis for data distribution (Histogram and Barcharts)
- Feature Selection based on data distribution
- Outlier treatment
- Missing Values treatment
- Visual correlation analysis
- Statistical correlation analysis (Feature Selection)
- Converting data to numeric for ML
- Sampling and K-fold cross validation
- Trying multiple Regression algorithms
- Selecting the best Model

- Deploying the best model in production

Looking at the distribution of Target variable

- If target variable's distribution is too **skewed** then the predictive modeling will not be possible.
- Bell curve is desirable but slightly positive skew or negative skew is also fine
- When performing Regression, make sure the histogram looks like a bell curve or slight skewed version of it. Otherwise it impacts the Machine Learning algorithms ability to learn all the scenarios.

Basic Data Exploration

This step is performed to guage the overall data. The volume of data, the types of columns present in the data. Initial assessment of the data should be done to identify which columns are Quantitative, Categorical or Qualitative.

This step helps to start the column rejection process. You must look at each column carefully and ask, does this column affect the values of the Target variable? For example in this case study, you will ask, **does this column affect the price of the house?** If the answer is a clear "No", then remove the column immediately from the data, otherwise keep the column for further analysis.

There are four commands which are used for Basic data exploration in Python

- **head()** : This helps to see a few sample rows of the data
- **info()** : This provides the summarized information of the data

- **describe()** : This provides the descriptive statistical details of the data
- **nunique()**: This helps us to identify if a column is categorical or continuous

Visual Exploratory Data Analysis

- Categorical variables: Bar plot
- Continuous variables: Histogram

Visualize distribution of all the Categorical Predictor variables in the data using bar plots

We can spot a categorical variable in the data by looking at the unique values in them. Typically a categorical variable contains less than 20 Unique values AND there is repetition of values, which means the data can be grouped by those unique values.

Based on the Basic Data Exploration above, we have spotted two categorical predictors in the data

Bar Charts Interpretation

These bar charts represent the frequencies of each category in the Y-axis and the category names in the X-axis.

In the ideal bar chart each category has comparable frequency. Hence, there are enough rows for each category in the data for the ML algorithm to learn.

If there is a column which shows too skewed distribution where there is only one dominant bar and the other categories are present in very low numbers. These

kind of columns may not be very helpful in machine learning. We confirm this in the correlation analysis section and take a final call to select or reject the column.

In this data, "CHAS" is skewed. There is just one bar which is dominating and other one have very less rows. Such columns may not be correlated with the target variable because there is no information to learn. The algorithms cannot find any rule like when the value is this then the target variable is that.

Whenever in doubt, always investigate the column further.

Histogram Interpretation

The ideal outcome for histogram is a bell curve or slightly skewed bell curve. If there is too much skewness, then outlier treatment should be done and the column should be re-examined, if that also does not solve the problem then only reject the column.

Outlier treatment

Outliers are extreme values in the data which are far away from most of the values. You can see them as the tails in the histogram.

Outlier must be treated one column at a time. As the treatment will be slightly different for each column.

Why I should treat the outliers?

Outliers bias the training of machine learning models. As the algorithm tries to fit the extreme value, it goes away from majority of the data.

There are below two options to treat outliers in the data.

- Option-1: Delete the outlier Records. Only if there are just few rows lost.
- Option-2: Impute the outlier values with a logical business value

Missing values treatment

Missing values are treated for each column separately.

If a column has more than 30% data missing, then missing value treatment cannot be done. That column must be rejected because too much information is missing.

There are below options for treating missing values in data.

- Delete the missing value rows if there are only few records
- Impute the missing values with MEDIAN value for continuous variables
- Impute the missing values with MODE value for categorical variables
- Interpolate the values based on nearby values
- Interpolate the values based on business logic

Feature Selection

Visual exploration of relationship between variables

- Continuous Vs Continuous ---- Scatter Plot
- Categorical Vs Continuous---- Box Plot
- Categorical Vs Categorical---- Grouped Bar Plots

Statistical measurement of relationship strength between variables¶

- Continuous Vs Continuous ---- Correlation matrix
- Categorical Vs Continuous---- ANOVA test
- Categorical Vs Categorical--- Chi-Square test

In this case study the Target variable is Continuous, hence below two scenarios will be present

- Continuous Target Variable Vs Continuous Predictor
- Continuous Target Variable Vs Categorical Predictor

Data Pre-processing for Machine Learning

List of steps performed on predictor variables before data can be used for machine learning

1. Converting each Ordinal Categorical columns to numeric
2. Converting Binary nominal Categorical columns to numeric using 1/0 mapping
3. Converting all other nominal categorical columns to numeric using pd.get_dummies()
4. Data Transformation (Optional): Standardization/Normalization/log/sqrt. Important if you are using distance based algorithms like KNN, or Neural Networks

Converting the ordinal variable to numeric

In this data there is no Ordinal categorical variable.

Converting the binary nominal variable to numeric using 1/0 mapping

There is no binary nominal variable in string format in this data

Converting the nominal variable to numeric using get_dummies(

Machine Learning: Splitting the data into Training and Testing sample

We dont use the full data for creating the model. Some data is randomly selected and kept aside for checking how good the model is. This is known as Testing Data and the remaining data is called Training data on which the model is built. Typically 70% of data is used as Training data and the rest 30% is used as Tesing data.

Standardization/Normalization of data

You can choose not to run this step if you want to compare the resultant accuracy of this transformation with the accuracy of raw dat

use Different algorithms to test accuracy :

1. `setup.py` is a python file, the presence of which is an indication that the module/package you are about to install has likely been packaged and

distributed with Distutils, which is the standard for distributing Python Modules

a. **Installation**: The primary purpose of a setup file is to install software onto a user's system. This involves copying files to appropriate directories
b. **Dependency Management**: Setup files can ensure that all necessary dependencies (libraries, frameworks, runtime environments, etc.) are installed on the user's system prior to installing the main software packag
c. setup function : it is used to share your code within the code enviroment.
d. python setup.py install : it will creates a library what ever package you have it will convert into lib , when whole project has been completed then you can do this to make it library
e.

It seems like you're referring to a

```
.egg-info
```

file, which is a metadata file commonly associated with Python packages installed via the setuptools library.

When you create and distribute a Python package, you often include metadata about the package itself. This metadata includes information such as the package name, version number, dependencies, author, license, and other details. This metadata is useful for package management tools and for users to understand the package's properties

conda create venv/ : for creating activating the conda

pip install -r requirements.txt to run requirements.txt file

-e . : do the same things what find packages will do as install the packges ,

-e . : it also editable install of a Python package. When you use `-e` in a `requirements.txt` file, it tells `pip` to install the package in "editable" mode, also known as development mode or "editable install".

`pip` installs the package in such a way that changes to the source code files immediately affect the installed package.

When you specify a package in editable mode in `requirements.txt` and then run `pip install -r requirements.txt`, `pip` installs the package in editable mode, allowing you to work on the package's source code directly without the need for reinstallation after each change

and it also trigger the setup,py file where we have written the code for the find paackages and install the package which is sensor , (means ab hm jarurat nhi setup.py file ko run krne ki automatic -e . usko run kr dega and package install ho jayege)

pipeline : it is sequence of steps that we have to take , create this folder inside sensor folder pipeline .

create pipeline folder inside it and also create **init** inside it .

component of of machine learning project are :

data access file : from mongodb bring data and read as a csv file for so create that code in deparate file

.

Amazon S3 (Simple Storage Service) is a scalable object storage service provided by Amazon Web Services (AWS). An S3 bucket is a container for storing objects, which can be virtually anything: from images, videos, and documents to application backups, software packages, and website files.

object storage vs file storage

cloud storage

configuration for makeing central connection between all the things

constant

logger and exception file

google mlops for component image

entity : for iinpt and output

artifacts : outpur of every component : all the tata and requred for model traning

ml functions : anything relate to custom to

Git:

1. **Git**:
    - Git is a distributed version control system (DVCS) used for tracking changes in source code during software development.
    - It allows developers to work collaboratively on projects, tracking changes, and managing versions of code.
    - With Git, developers can create branches, merge changes, revert to previous states, and collaborate seamlessly on codebases.
    - Git operates locally on a developer's machine, allowing them to work offline and commit changes to their local repository.
2. **GitHub**:
    - GitHub is a web-based platform that provides hosting for Git repositories.
    - It offers additional features on top of Git, such as issue tracking, pull requests, code review, project management tools, and wikis.
    - GitHub allows developers to store their Git repositories remotely in the cloud, making it easy to share code and collaborate with others.
    - It provides a graphical user interface (GUI) for managing repositories, as well as APIs for automation and integration with other tools.

tracking : status to check whether track or not

git add to do tracking

staging : preaptre chages to be commited

git commit prepare thing of push

git push to sending the changes

In Git, the "M" sign next to a file indicates that the file has been modified in your working directory. This means you have made changes to the file since the last commit.

The "A" sign next to a file indicates that the file has been added to the staging area. This means you have used the `git add` command to stage the changes in the file, preparing them to be included in the next commit.

git config

git init

git status

git add .

git commit

git remote add origin

git push origin main

git branch

git checkout developer2

git ranch develper1

git merge developer1

git log

git show

Git is a distributed version control system widely used for tracking changes in source code during software development. Below, I'll provide an overview of some key concepts in Git along with examples of commonly used commands:

1. **Repository (Repo)**:
   - A repository is a directory or storage space where your project files are stored, along with metadata about the project's history.
   - Example command to initialize a new Git repository:

```
csharpCopy code
git init
```

2. **Commit**:
   - A commit is a snapshot of changes made to the repository at a specific point in time.
   - Example command to commit changes:

```
sqlCopy code
git commit -m "Commit message"
```

3. **Branch**:
   - A branch is a separate line of development that diverges from the main line (often called the "master" or "main" branch).
   - Example command to create a new branch:

```
arduinoCopy code
git branch new-feature
```

4. **Merge**:
    - Merging combines changes from different branches into a single branch, typically the main branch.
    - Example command to merge branches:

```
sqlCopy code
git merge new-feature
```

5. **Remote**:
    - A remote is a common repository that all team members use to exchange their changes.
    - Example command to add a remote repository:

```
csharpCopy code
git remote add origin <remote-url>
```

6. **Pull**:
    - Pulling fetches changes from a remote repository and integrates them into the current branch.
    - Example command to pull changes from a remote repository:

```
cssCopy code
git pull origin main
```

7. **Push**:
    - Pushing sends local commits to a remote repository.
    - Example command to push changes to a remote repository:

```
cssCopy code
git push origin main
```

8. **Clone**:
    - Cloning creates a copy of a repository from a remote source to your local machine.
    - Example command to clone a repository:

```
bashCopy code
git clone <remote-url>
```

9. **Status**:
    - Status shows the current status of the repository, including modified, staged, and untracked files.
    - Example command to check the status of the repository:

```
luaCopy code
git status
```

10. **Log**:
    - Log displays a list of commits in reverse chronological order.
    - Example command to view the commit history:

```
bashCopy code
git log
```

These are just some of the fundamental concepts and commands in Git. Git offers a wide range of functionalities and commands for version control,

collaboration, and project management. As you become more familiar with Git, you'll disc

Certainly! Here are some more Git commands related to various steps in the software development workflow:

1. **Branch Operations**:
   - Create a new branch: `git branch <branch-name>`
   - Switch to a different branch: `git checkout <branch-name>`
   - Create and switch to a new branch: `git checkout -b <branch-name>`
   - Delete a branch: `git branch -d <branch-name>`
   - Rename a branch: `git branch -m <new-branch-name>`
2. **Staging Changes**:
   - Add changes to the staging area: `git add <file-name>`
   - Add all changes to the staging area: `git add .`
   - Remove files from the staging area: `git reset <file-name>`
   - Unstage all changes: `git reset`
3. **Comparing Changes**:
   - Show changes between commits, branches, etc.: `git diff`
   - Show changes in staged files: `git diff --staged`
4. **Undoing Changes**:
   - Discard changes in a file (unstaged): `git checkout -- <file-name>`
   - Discard all changes in working directory: `git reset --hard`
   - Undo the last commit (keep changes in working directory): `git reset HEAD~`
   - Undo the last commit and discard changes: `git reset --hard HEAD~`

5. **Viewing History**:
    - Show commit history with commit IDs: `git log`
    - Show abbreviated commit IDs and commit messages: `git log --oneline`
    - Show detailed commit information: `git show <commit-id>`
6. **Remote Repository**:
    - Fetch changes from a remote repository: `git fetch`
    - View remote branches: `git branch -r`
    - Fetch and merge changes from a remote repository: `git pull`
    - Push changes to a remote repository: `git push <remote-name> <branch-name>`
    - View remote repository URLs: `git remote -v`
7. **Collaboration**:
    - Show changes introduced by others: `git fetch` followed by `git diff origin/<branch-name>`
    - Resolve merge conflicts: Manually edit conflicted files, then `git add` and `git commit`
    - Create a pull request (GitHub/GitLab): `git push origin <branch-name>` followed by creating a pull request in the web interface

when you do and changes in git directly :

git fetch origin

1. **Rebase or Merge Changes**:
   Once you've fetched the updates from the remote repository, you have two options:
    - **Rebase**: If you prefer to apply your local changes on top of the remote changes, you can use rebase:

```
bashCopy code
git rebase origin/main
```

- **Merge**: If you prefer to merge the remote changes into your local branch, you can use merge:

```
bashCopy code
git merge origin/main
```

git push origin main

error 2:git pull —rebase origin main

Error3 :

- Use `git pull` to sync your local branch with the latest changes from the remote repository, especially when working on shared branches.
- Use `git rebase` to incorporate changes from one branch onto another in a cleaner and more linear manner, especially when maintaining feature branches or collaborating on shared branches.

Q : remote origin laredy exist

git remote rm origin

git remote add origin

git remote -v

git remote remove origin

git remote add <new-remote-name> <remote-repository-url>

QEverything up to date

Q . no origin found over there

Qpath epscisific error :

Q authetication errror

class 2 : kafka :

1.conda —version for checking version

dir : for checking the directory

ls : for checking the directory

create virtual env -. conda create -p venv python ==3.8

always try to save the code to remote error like : sensor as module not found

conda activate venv/

python setup.py install

pip3 install -r requirements.txt

#for those who donot have conda configure#

conda —veersion : to. check the version

how to set env :

serch environment , edit envirment , in user edit and and set path