MACHINE
LEARNING

FLIGHT PRICE PREDICTION

PREDICTING FLIGHT FARE USING MACHINE LEARNING

OBJECTIVE

TO DEVELOP A MACHINE LEARNING ALGORITHM THAT CAN PREDICT THE FLIGHT FARE FOR A GIVEN DATE BETWEEN CERTAIN CITIES

BENEFITS

- REDUCE TRAVEL COSTS
- GIVES BETTER INSIGHT OF FLIGHT FARE CHANGES
- HELPS TO MAKE BETTER TRAVEL PLANS

DATASET

- This dataset is from kaggle website
- Dataset is in CSV format , so we have used pandas for dataset manipulations Dataset contains both numerical and categorical data , so we have also used some data transformation techniques in our dataset
- We have data of date, class, flight code , day of the week, Airline , Source , destination, departure time , arrival time , total stops between source and destination , duration of flight , days left and fare
- Some columns should be transformed and some columns should be avoided , all these operations are explained in detail

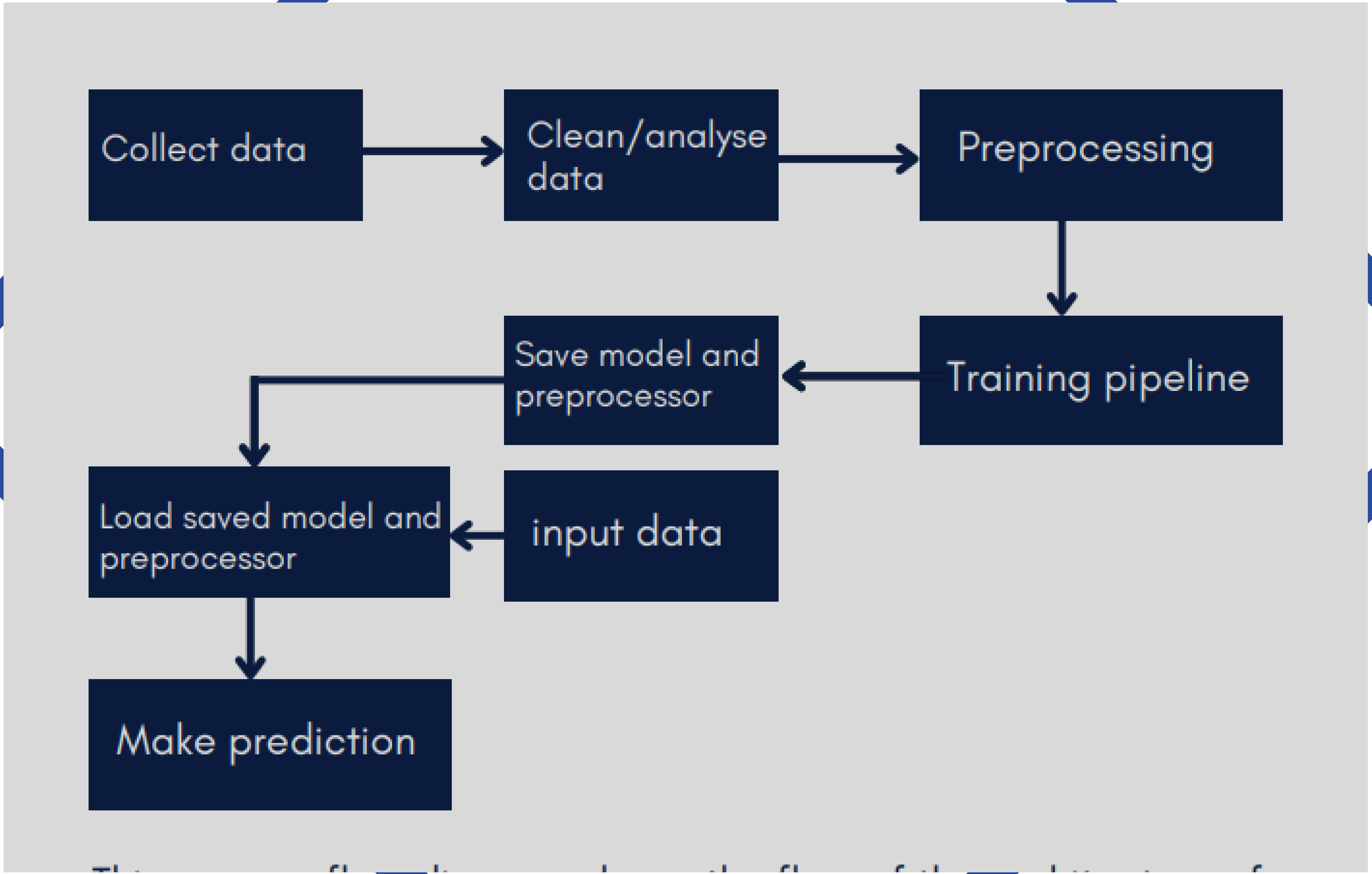
Model training

- For training first we split the dataset into X and y . X is our features and y is our target variable
- we have used "train test split" from **scikit-learn** library
- We have developed a pipeline using **pipeline** module from scikit learn. we have implemented a column transformer that can transform categorical columns using one hot encoder we will save the preprocessor to re use on the new dataframe . we will use the pickle library to save the **preprocessor**
- to train the model , we have tested some regression algorithms to check which model will give us the best score. we ended up on the **random forest regressor** .

TOOLS

- **PYTHON** FOR CORE CODING
- **FLASK FRAMEWORK** FOR WEB APP UI
- **PANDAS** FOR DATASET OPS
- **SCIKIT-LEARN** FOR MACHINE LEARNING
- **VS CODE** AS IDE

Architecture



Error handling

To handle the errors , we have made the **CustomException** class

That can be called in a **try-except** block to log the exact error with file name and line number In Python

The **try-except** block is used to handle exceptions that may occur during the execution of our code.

Logging

We have used python **loggings** to log the checkpoints throughout the program

it creates a folder called "**logs**" and then stores all the logs inside the folder with timestamp

So we can always check out the flow of the program

Accuracy

we have saved the preprocessor and model with an R-square value of 0.90 !

We can check the accuracy of our model and we have done it in our notebook

although in practical case, we can just go to any travel websites and check if our prediction is nearly accurate