Project

Predict Fare of Airlines Tickets

(1) Data preprocessing

→ Dealing with missing values (null) { Using dropna }

→ Converting the 'object' datatype to 'Date-Time format' for the ~~columns~~ required columns like 'Arrival Time', 'Dept. Time'

* User-defined function to convert 'object' to Date-time format (OR any)

```
def change_into_datetime(column):
    train_df[column] = pd.to_datetime[column]
```

```
def change_into_datetime(col):
    train_df[col] = pd.to_datetime(train_df[col])
```

(To convert multiple columns run loop with i for all column names.

→ Splitting the timestamp into (Date) | (Month) | (Year)

→ To extract hour, minutes from Departure / Arrival Time columns.

function

```
def extract_hour(train_df, col):
    train_df[col].dt.hour
    OR
    train_df[col + '_hour'] = train_df[col].dt.hour
```

Camlin | Page
Date / /

Same function of time

```
def extract_min(train_df, col):
    train_df[col].dt.
    train_df[col + '_minute'] = train_df[col].dt.minute
```

→ In 'Duration' column, we have to iterate the values of hour and minutes.

Where there is no minute add '0m' and where there is no hour, add '0h'.

Eg → 2h 5m ⟶ will make a list → '2h', '5m'.
1 2

⟶ Delimiter is ' ' (space).

```
duration = list(train_df['Duration'])
for i in range(len(duration)):
    if len(duration[i].split(' ')) == 2:
        pass
    else:
        if 'h' in duration[i]:
            duration[i] = duration[i] + ' 0m'   # space
        else:
            duration[i] = '0h ' + duration[i]
```

Now to extract only numeric value in 'duration' list we will use split

Eg '2h' '5m'
[0][1] [0][1]
[0] [1]

∴ To extract 2 :- split(' ')[0][0:-1]

∴ To extract 5 :- split(' ')[1][0:-1]

Put these logic in functions.

→ To differentiate between categorical data columns and numeric data columns.

Following statement is a code of "List Comprehension" in Python.

```
cat_col = [col for col in train_df.columns if
                train_df[col].dtype == 'O']
print(cat_col)
```

[Out] → All column names with categorical data in Dataframe

(This means,)

col (for col in train_df.columns) if train_df[col].dtype == 'O'

Iteration part — Condition part

dtype 'O' means 'object' type

If condition == true then only column will be column will be considered in list (output)

→ Dealing with categorical data

Categorical data

↓ Nominal Data

(Those data that have no Hierarchy/Order)

↓ One Hot Encoding To be done

↓ Ordinal Data

(Those data that have some kind of Hierarchy)

↓ Label encoding To be Done

Camlin | Page
Date / /

Nominal Data → One Hot Encoding
Ordinal Data → Label Encoding

→ Visualization of 'Airlines' using boxplot.
and
('Total_stop')

→ Creating Dummy variables (Doing One Hot Encoding) on categorical columns → Airlines, Source, Destiny

Set (drop_first = True) ← This will create (k-1) dummy variables, if k is the total count.

↳ This parameter helps to reduce correlations created among the dummy variables.

→ 'Route column' - Split the city codes like BLR, DEL using delimiter '→' then put them in different columns.

Perform label Encoding on all the columns.

Replace the null values using 'None'

→ 'Total_stops' column : i) Find unique values in column.

ii) Put them in dictionary and map it to replace it with values → 0, 1, 2, 3, 4 → {'non-stop':0, '2 stops':2, '1 stop',

See this # Do not change Order, else ERROR

(2) How to Deal with Outliers

→ (i) Distribution approach
(ii) Box plot approach

Function to display the distribution plot and the box plot.

```
def plot(df, col):
    fig, (ax1, ax2) = plt.subplots(2, 1)
    sns.distplot(df[column], ax = ax1)
    sns.boxplot(df[column], ax = ax2)
```

This function helps to get the outliers position.

(3.) Separating out dependent and independent feature.

X	Y
dependent features (All except 'Price')	Independent feature 'Price'

Y is to be predicted using X features.

(4) To find best features.

Use mutual_info_classif module to get the importance count of the columns.

(5) Apply Train Test Split.

Camlin Page
Date / /

(6). Use Random Forest Regressor → 81% ✱ Best algo

(7). Use Linear Regression → 63%

(8). Use KNNs → 68%

(9). Use Decision Tree → 69%

(10). Perform ~~Hyperparameter~~ Hyperparameter Tuning.

Declaring n_estimators and max_depths.

```
n_estimators = [int(x) for x in np.linspace(start=100, stop=1200, num=6)]
max_depth = [int(x) for x in np.linspace(start=5, stop=30, num=4)]
```

→ Creating dictionary to feed the parameters to the RandomizedSearchCV.

```
random_params {
        'n_estimators': n_estimators
        'max_features': ['auto', 'sqrt']
        'max_depth': max_depth
        'min_samples_split': [5, 10, 15, 100]
    }.
```

(11). Use Hypertuned ~~Retrofit~~ Random Forest Regressor Algo → 84%.

(12). Save model locally using pickle.

(13). Load Testing Data → ~~Re~~ Load Prebrained Model → Preprocess Test Data → Perform Predictions → Save the new xlsx file locally.

X