In [1]:
```
pip install gensim
```

```
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: https://anu9rng:****@rb-artifactory.bosch.com/artifactory/api/pypi/python-virtual/simple
Requirement already satisfied: gensim in c:\users\usm8kor\appdata\roaming\python\python39\site-packages (4.3.3)
Requirement already satisfied: numpy<2.0,>=1.18.5 in c:\users\usm8kor\appdata\roaming\python\python39\site-packages (from gensim) (1.23.5)
Requirement already satisfied: scipy<1.14.0,>=1.7.0 in c:\users\usm8kor\appdata\roaming\python\python39\site-packages (from gensim) (1.13.1)
Requirement already satisfied: smart-open>=1.8.1 in c:\users\usm8kor\appdata\roaming\python\python39\site-packages (from gensim) (7.0.4)
Requirement already satisfied: wrapt in c:\program files\anaconda3\lib\site-packages (from smart-open>=1.8.1->gensim) (1.12.1)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import pandas as pd

# Load the dataset
file_path = 'Data_Train.xlsx'
df = pd.read_excel(file_path)

df.head()
```

Out[2]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

Column-wise Transformations:

```
Airline:
    Transformation: One-Hot Encoding (OHE) within categorical_pipeline
Date_of_Journey:
    Transformation: Split into Journey_Day and Journey_Month (then dropped)
Source:
    Transformation: TF-IDF Vectorization within tfidf_pipeline
Destination:
    Transformation: TF-IDF Vectorization within tfidf_pipeline
Route:
    Transformation: One-Hot Encoding (OHE) within categorical_pipeline
Dep_Time:
    Transformation: Split into Dep_Hour and Dep_Minute (then dropped)
Arrival_Time:
    Transformation: Split into Arrival_Hour and Arrival_Minute (then dropped)
Duration:
    Transformation: Converted to Duration_Minutes (then dropped)
Total_Stops:
    Transformation: Label Encoding
Additional_Info:
    Transformation: Word2Vec Embedding into Additional_Info_Word2Vec (then dropped)
Price:
    Transformation: Target Column (No Transformation)
```

Additional Information:

```
Numerical Columns (Journey_Day, Journey_Month, Dep_Hour, Dep_Minute, Arrival_Hour, Arrival_Minute, Duration_Minutes, Total_Stops):
    Transformation: Imputation with SimpleImputer, followed by scaling with StandardScaler within numerical_pipeline.
Categorical Columns (Airline, Route):
    Transformation: Imputation with SimpleImputer, followed by One-Hot Encoding within categorical_pipeline.
```

In [3]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler, LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
from sklearn.feature_extraction.text import TfidfVectorizer
from gensim.models import Word2Vec
import numpy as np

# Feature Engineering: Splitting Date_of_Journey into day, month, year
df['Journey_Day'] = pd.to_datetime(df['Date_of_Journey'], format='%d/%m/%Y').dt.day
df['Journey_Month'] = pd.to_datetime(df['Date_of_Journey'], format='%d/%m/%Y').dt.month

# Feature Engineering: Extracting hours and minutes from Dep_Time and Arrival_Time
df['Dep_Hour'] = pd.to_datetime(df['Dep_Time']).dt.hour
df['Dep_Minute'] = pd.to_datetime(df['Dep_Time']).dt.minute
df['Arrival_Hour'] = pd.to_datetime(df['Arrival_Time']).dt.hour
df['Arrival_Minute'] = pd.to_datetime(df['Arrival_Time']).dt.minute

# Feature Engineering: Splitting Duration into hours and minutes
duration_split = df['Duration'].str.extract(r'(?:(\d+)h)?\s*(?:(\d+)m)?')
df['Duration_Minutes'] = duration_split[0].fillna(0).astype(int) * 60 + duration_split[1].fillna(0).astype(int)

# Label Encoding for Total_Stops
label_encoder = LabelEncoder()
df['Total_Stops'] = label_encoder.fit_transform(df['Total_Stops'])

# Word2Vec for Additional_Info
additional_info_sentences = df['Additional_Info'].apply(lambda x: x.split()).tolist()
word2vec_model = Word2Vec(sentences=additional_info_sentences, vector_size=50, window=3, min_count=1, workers=4)
df['Additional_Info_Word2Vec'] = df['Additional_Info'].apply(
    lambda x: np.mean([word2vec_model.wv[word] for word in x.split() if word in word2vec_model.wv], axis=0)
    if len(x.split()) > 0 else np.zeros(50)
)

# Dropping columns that are no longer needed
df.drop(['Date_of_Journey', 'Dep_Time', 'Arrival_Time', 'Duration', 'Additional_Info'], axis=1, inplace=True)

# Preprocessing for categorical data and feature scaling
categorical_cols = ['Airline', 'Route']
```

```
numerical_cols = ['Journey_Day', 'Journey_Month', 'Dep_Hour', 'Dep_Minute', 'Arrival_Hour', 'Arrival_Minute', 'Duration_Minutes', 'Total_Stops']
word2vec_col = ['Additional_Info_Word2Vec']

# Pipeline for categorical features (with OneHotEncoder)
categorical_pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
])

# Pipeline for Source and Destination (with TF-IDF Vectorizer)
tfidf_pipeline = Pipeline(steps=[
    ('tfidf', TfidfVectorizer())
])

# Pipeline for numerical features
numerical_pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

# ColumnTransformer to apply transformations to appropriate columns
preprocessor = ColumnTransformer(transformers=[
    ('num', numerical_pipeline, numerical_cols),
    ('cat', categorical_pipeline, categorical_cols),
    ('tfidf_src', TfidfVectorizer(), 'Source'),
    ('tfidf_dest', TfidfVectorizer(), 'Destination')
])

# Defining the pipeline with preprocessing and model
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', LinearRegression())
])

# Splitting the data into training and test sets
X = df.drop('Price', axis=1)
y = df['Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Fitting the model
pipeline.fit(X_train, y_train)

# Evaluating the model
score = pipeline.score(X_test, y_test)
print(f"Model R^2 score: {score}")

# Checking the transformation on the first row
X_transformed = pipeline.named_steps['preprocessor'].transform(X.head(1))

# Debugging: Print the transformed output
print(f"Transformed output:\n{X_transformed}")

# Check the shape of the transformed data
print(f"Shape of transformed data: {X_transformed.shape}")
```

```
Model R^2 score: -4.689424679936029e+16
Transformed output:
[[ 1.22840525 -1.46907017  1.65637489 -0.23582949 -1.79939492 -0.89024038
  -0.92856352  1.40623786  0.          0.          0.          1.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          1.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.
   0.          1.          0.          0.          0.          0.
   0.          0.          0.60090709  0.          0.          0.79931888]]
Shape of transformed data: (1, 156)
```

In [ ]: