Price Prediction using Regression

This is a tickets pricing monitoring system. It scrapes tickets pricing data periodically and stores it in a database. Ticket pricing changes based on demand and time, and there can be significant difference in price. We are creating this product mainly with ourselves in mind. Users can set up alarms using an email, choosing an origin and destination (cities), time (date and hour range picker) choosing a price reduction over mean price, etc.

Following is the description for columns in the dataset

- insert_date: date and time when the price was collected and written in the database
- origin: origin city
- destination: destination city
- start_date: train departure time
- end_date: train arrival time
- train_type: train service name
- price: price
- train_class: ticket class, tourist, business, etc.
- fare: ticket fare, round trip, etc

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
sns.set()
```

Task 1: Import Dataset and create a copy of that dataset

```
#write code here
data = pd.read_csv('data1.csv')
df = data.copy()
```

Task 2: Display first five rows

```
#write code here
df.head()
```

	Unnamed: 0	insert_date	origin	destination	start_date	end_date	train_type	price	train_class	fare
0	702	2019-04-19 05:37:35	PONFERRADA	MADRID	2019-06-02 15:00:00	2019-06-02 19:42:00	MD-AVE	59.50	Turista con enlace	Flexible
1	703	2019-04-19 05:37:35	PONFERRADA	MADRID	2019-06-02 17:15:00	2019-06-02 23:03:00	MD-AVE	34.65	Turista con enlace	Promo +
2	704	2019-04-19 05:37:35	PONFERRADA	MADRID	2019-06-02 17:15:00	2019-06-02 23:10:00	MD-LD	39.95	Turista con enlace	Promo +
3	705	2019-04-19 05:37:35	PONFERRADA	MADRID	2019-06-02 17:15:00	2019-06-02 22:14:00	MD-AVE	40.60	Turista con enlace	Promo +
4	706	2019-04-19 05:37:35	PONFERRADA	MADRID	2019-06-02 18:55:00	2019-06-02 23:03:00	ALVIA	27.90	Turista	Promo

Task 3: Drop 'unnamed: 0' column

```
#write code here
df.drop('Unnamed: 0', axis=1, inplace=True)
```

Task 4: Check the number of rows and columns

```
#write code here
df.shape
```

```
(215909, 9)
```

Task 5: Check data types of all columns

```
#write code here
df.dtypes
```

```
insert_date     object
origin          object
destination     object
start_date      object
end_date        object
train_type      object
price          float64
train_class     object
fare            object
dtype: object
```

Task 6: Check summary statistics

```
#write code here
df.describe()
```

	price
count	202321.000000
mean	56.723877
std	25.531787
min	16.600000
25%	28.350000
50%	53.400000
75%	76.300000
max	206.800000

Task 7: Check summary statistics of all columns, including object dataypes

```
df.describe(include="all")
```

	insert_date	origin	destination	start_date	end_date	train_type	price	train_class	fare
count	215909	215909	215909	215909	215909	215909	202321.000000	215266	215266
unique	30543	5	5	2231	2870	16	NaN	5	5
top	2019-05-09 15:02:02	MADRID	MADRID	2019-06-02 17:30:00	2019-06-02 23:03:00	AVE	NaN	Turista	Promo
freq	90	110440	105469	2089	1278	126577	NaN	164016	132065
mean	NaN	NaN	NaN	NaN	NaN	NaN	56.723877	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	25.531787	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	16.600000	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	28.350000	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	53.400000	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	76.300000	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	206.800000	NaN	NaN

Question: Explain the summary statistics for the above data set

Answer: From this summary statistics we conclude that there are total 215909 set of rows from which some rows of price, train_class and fare are empty or contain null values. Also average price for ticket is 56 and max price for ticket is 209. There are 5 unique values for both origin and destination which means there are total 5 stations available in given dataset

Task 8: Check null values in dataset

```
#write code here
df.isnull().sum()
```

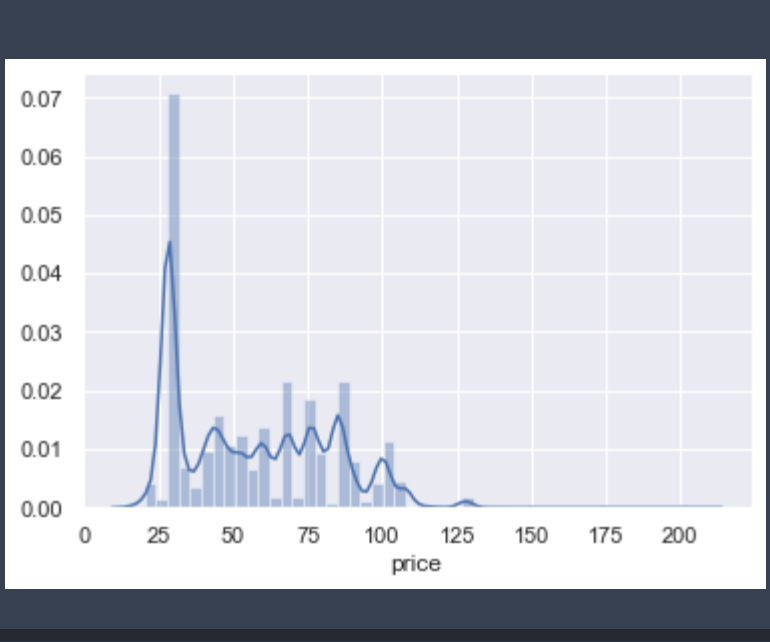
```
insert_date        0
origin             0
destination        0
start_date         0
end_date           0
train_type         0
price          13588
train_class      643
fare             643
dtype: int64
```

Task 9: Fill the Null values in the 'price' column.

Data in price colomn is right skewed so we replace null values with median

```
sns.distplot(df['price'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x653f025e08>
```




```
#write code here
df['price'].fillna(df['price'].median(),inplace=True)
```

Task 10: Drop the rows containing Null values in the attributes train_class and fare

```
#write code here
df.dropna(subset=['train_class','fare'],inplace=True)
```

Task 11: Drop 'insert_date'

```
#write code here
df.drop('insert_date', axis=1, inplace=True)
df.shape
```

```
(215266, 8)
```

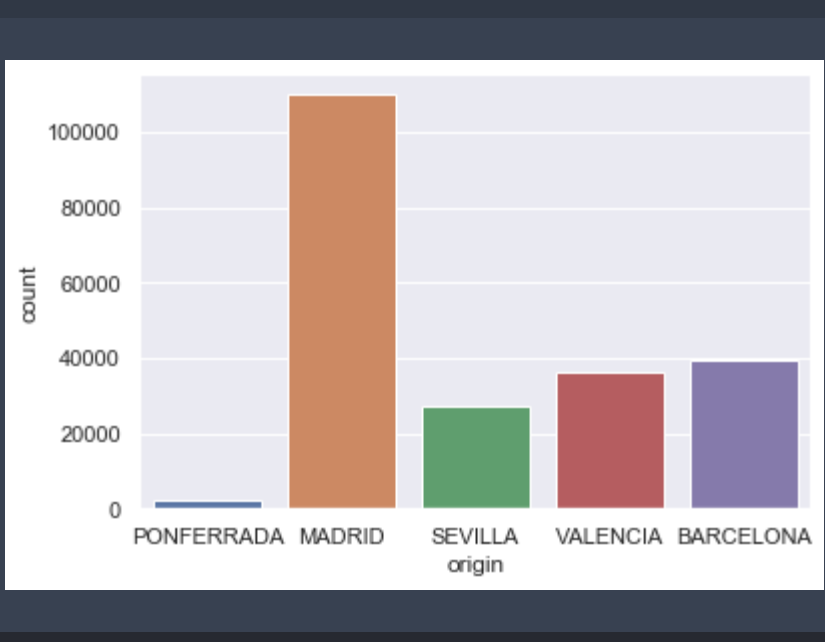
Check null values again in dataset

```
#write code here
df.isnull().sum()
```

```
origin         0
destination    0
start_date     0
end_date       0
train_type     0
price          0
train_class    0
fare           0
dtype: int64
```

Task 12: Plot number of people boarding from different stations

```
#write code here
sns.countplot(x='origin', data=df);
```

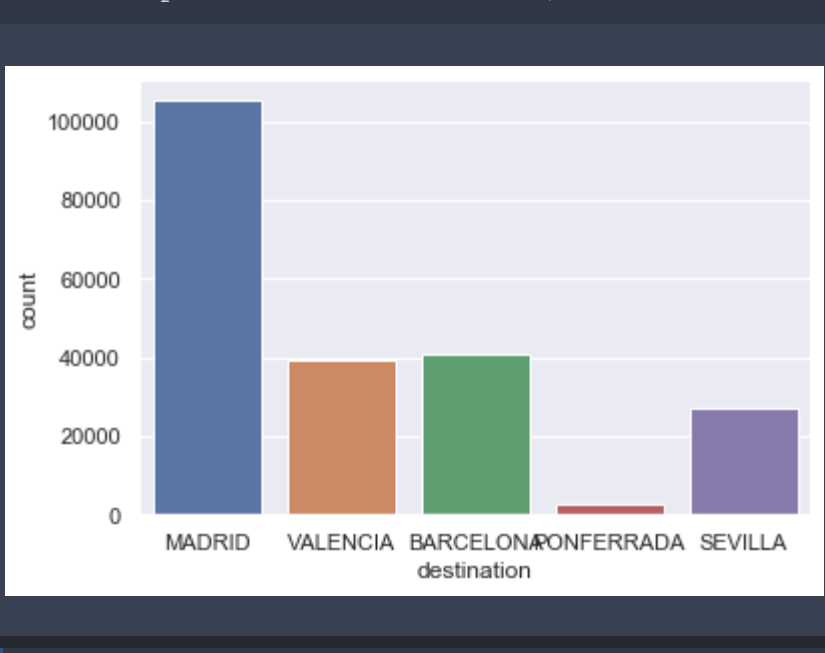


Question: What insights do you get from the above plot?

Answer: Most common station for boarding is madrid, around 150,000 or more people board from this station. Ponferrada have very less number of people boarding as compared to other stations. After madrid Barcelona have more number of boardings.

Task 13: Plot number of people for the destination stations

```
#write code here
sns.countplot(x='destination', data=df);
```



Question: What insights do you get from the above graph?

Answer: Madrid is the most common station used for travelling there were also huge number of people boarding from this station and as well as huge number of people getting off this station and very less number of people are getting off from Ponferrda station

Task 14: Plot different types of train that runs in Spain

```
#write code here
#sns.countplot(x='train_type', data=df)
plt.figure(figsize=(15,6))
ax = sns.countplot(x='train_type', data=df)
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
plt.tight_layout()
plt.show()
```

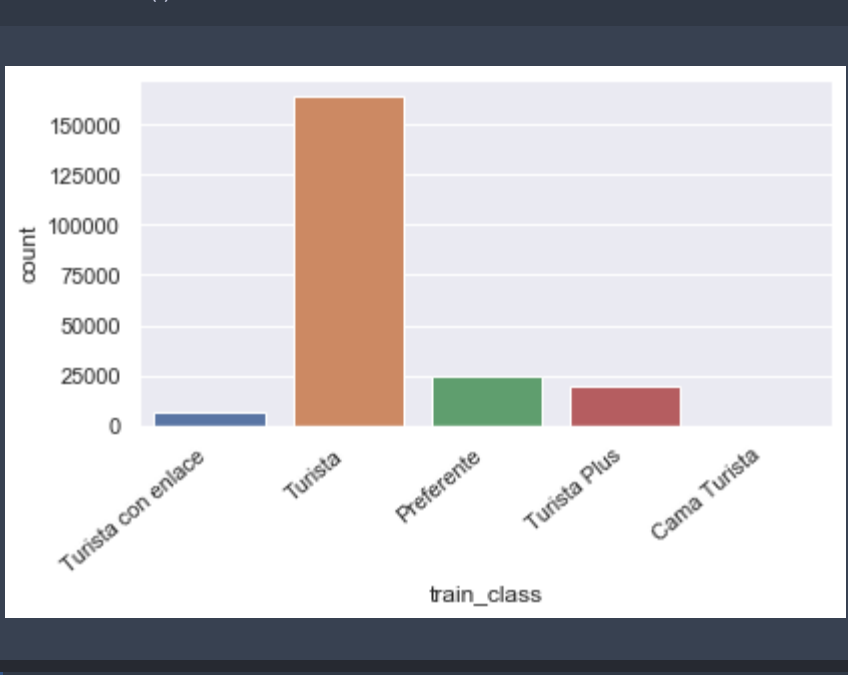
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-6046c305afb9> in <module>
      1 #write code here
      2 #sns.countplot(x='train_type', data=df)
----> 3 plt.figure(figsize=(15,6))
      4 ax = sns.countplot(x='train_type', data=df)
      5 ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")

NameError: name 'plt' is not defined
```

Question: Which train runs the maximum in number as compared to other train types?

Answer: Ave train type in spain have maximum number as compared to other train type

Task 15: Plot number of trains of different class

```
#write code here
#plt.figure(figsize=(15,6))
ax = sns.countplot(x='train_class', data=df)
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
plt.tight_layout()
plt.show()
```



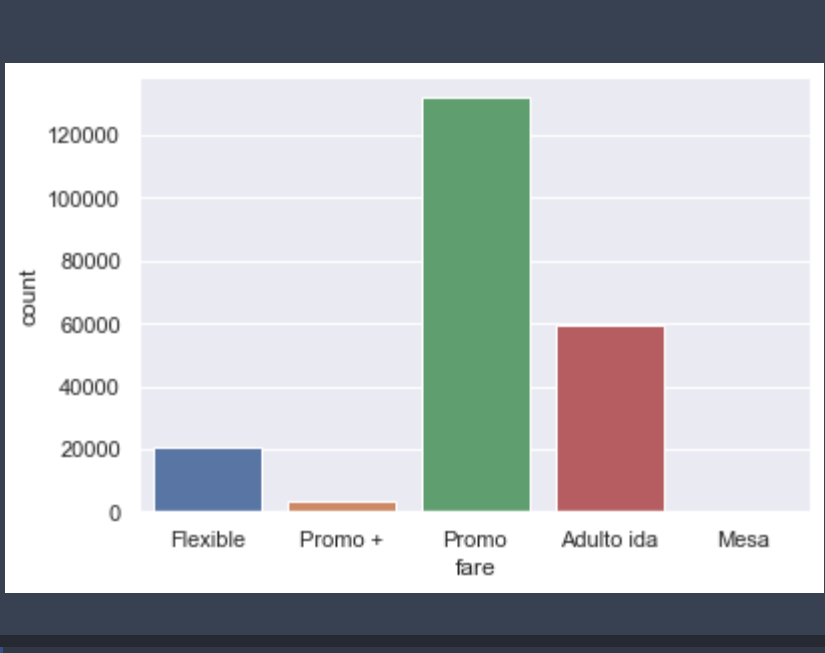
Question: Which the most common train class for traveling among people in general?

Answer:
Tunista is most common train class for traveling ammong people in spain.

Task 16: Plot number of tickets bought from each category

```
#write code here
sns.countplot(x='fare', data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x6539cd5c88>
```



Question: Which the most common tickets are bought?

Answer: Promo tickets are the most common ticket bought among people

Task 17: Plot distribution of the ticket prices

```
#write code here
sns.distplot(df['price'])
```

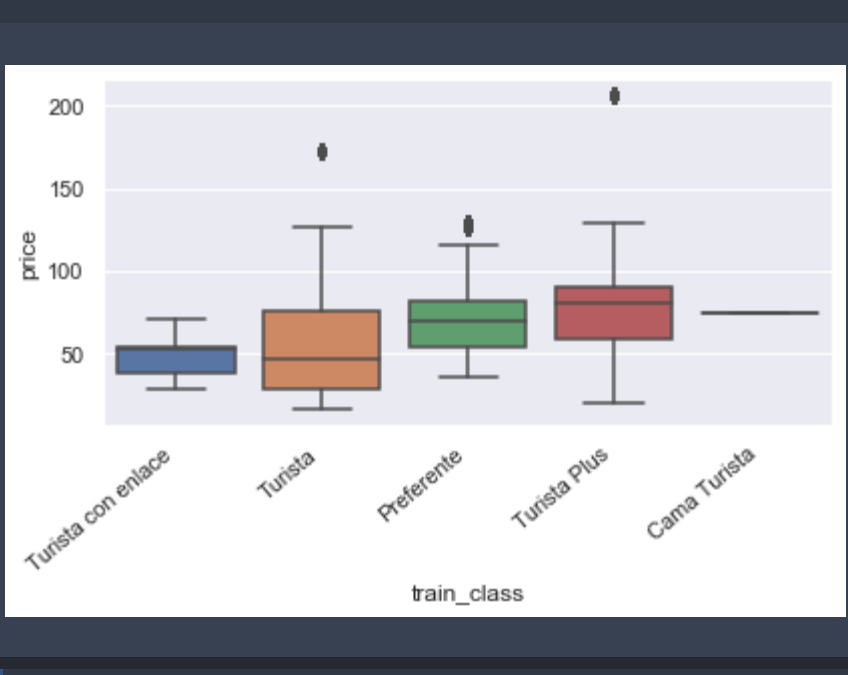
```
<matplotlib.axes._subplots.AxesSubplot at 0x653415ea88>
```

Question: What readings can you get from the above plot?

Answer:
From this distribution it looks like that price is positively skewed

Task 18: Show train_class vs price through boxplot

```
#write code here

ax = sns.boxplot(x = 'train_class', y = 'price', data = df)
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
plt.tight_layout()
plt.show()
```

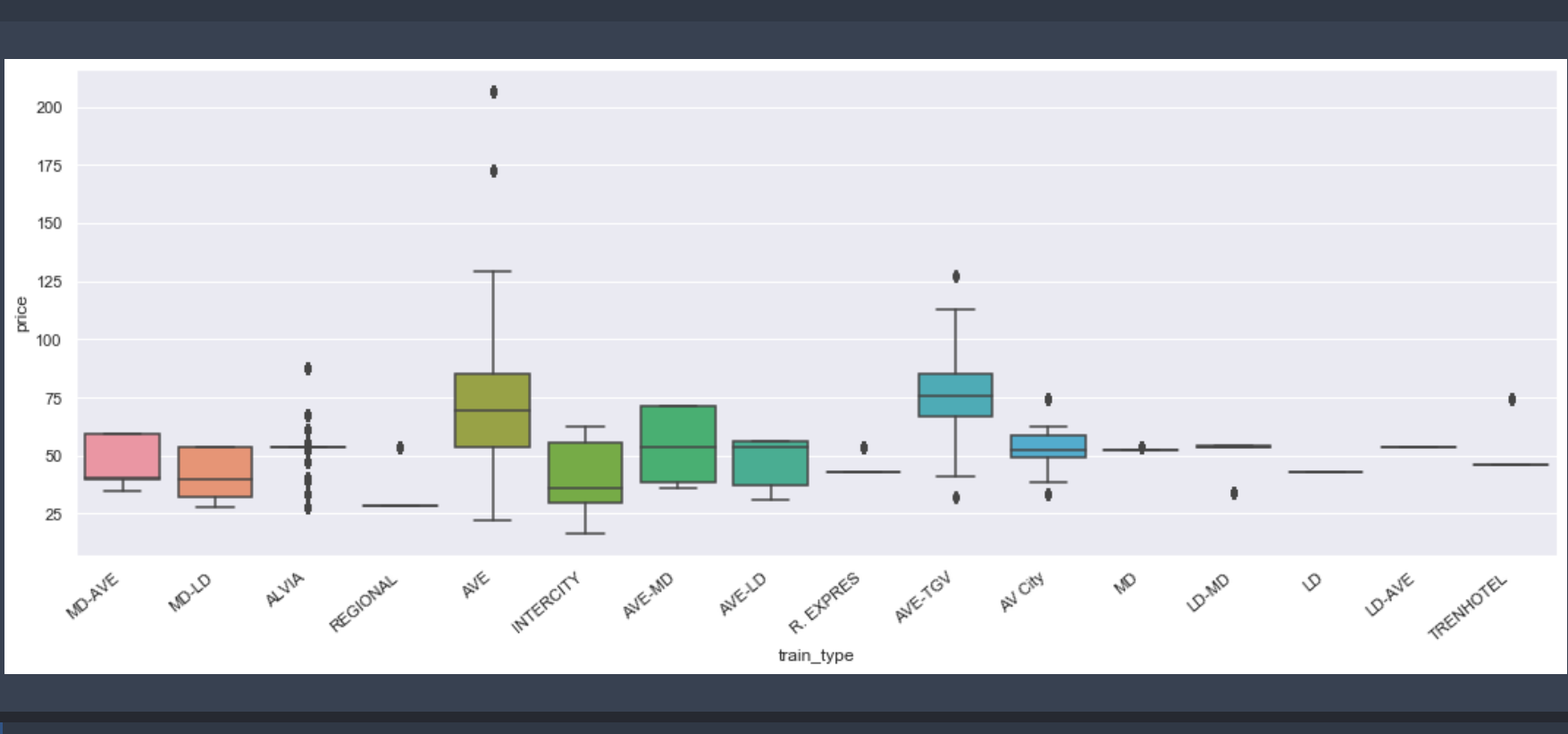


Question: What pricing trends can you find out by looking at the plot above?

Answer: Most of the ticket bought for class type range from 40 to 80. preferente class type have normal distribution among price and tunista is positively skewed and tunista plus is negatively skewed

Task 19: Show train_type vs price through boxplot

```
#write code here
plt.figure(figsize=(15,6))
ax = sns.boxplot(x = 'train_type', y = 'price', data = df)
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="center")
plt.tight_layout()
plt.show()
```



Question: Which type of trains cost more as compared to others?

Answer: Ave train type cost more as compared to other train type.

Feature Engineering

```
df = df.reset_index()
```

Finding the travel time between the place of origin and destination
We need to find out the travel time for each entry which can be obtained from the 'start_date' and 'end_date' column. Also if you see, these columns are in object type therefore datetimeFormat should be defined to perform the necessary operation of getting the required time.

Import datetime library

```
#write code here
import datetime
```

```
datetimeFormat = '%Y-%m-%d %H:%M:%S'
def fun(a,b):
    diff = datetime.datetime.strptime(b, datetimeFormat)- datetime.datetime.strptime(a, datetimeFormat)
    return(diff.seconds/3600.0)
```

```
df['travel_time_in_hrs'] = df.apply(lambda x:fun(x['start_date'],x['end_date']),axis=1)
```

Task 20: Remove redundant features

You need to remove features that are giving the related values as 'travel_time_in_hrs'
Hint: Look for date related columns

```
#write code here
df.drop(['start_date', 'end_date'], axis=1, inplace=True)
```

We now need to find out the pricing from 'MADRID' to other destinations. We also need to find out time which each train requires for travelling.

Travelling from MADRID to SEVILLA

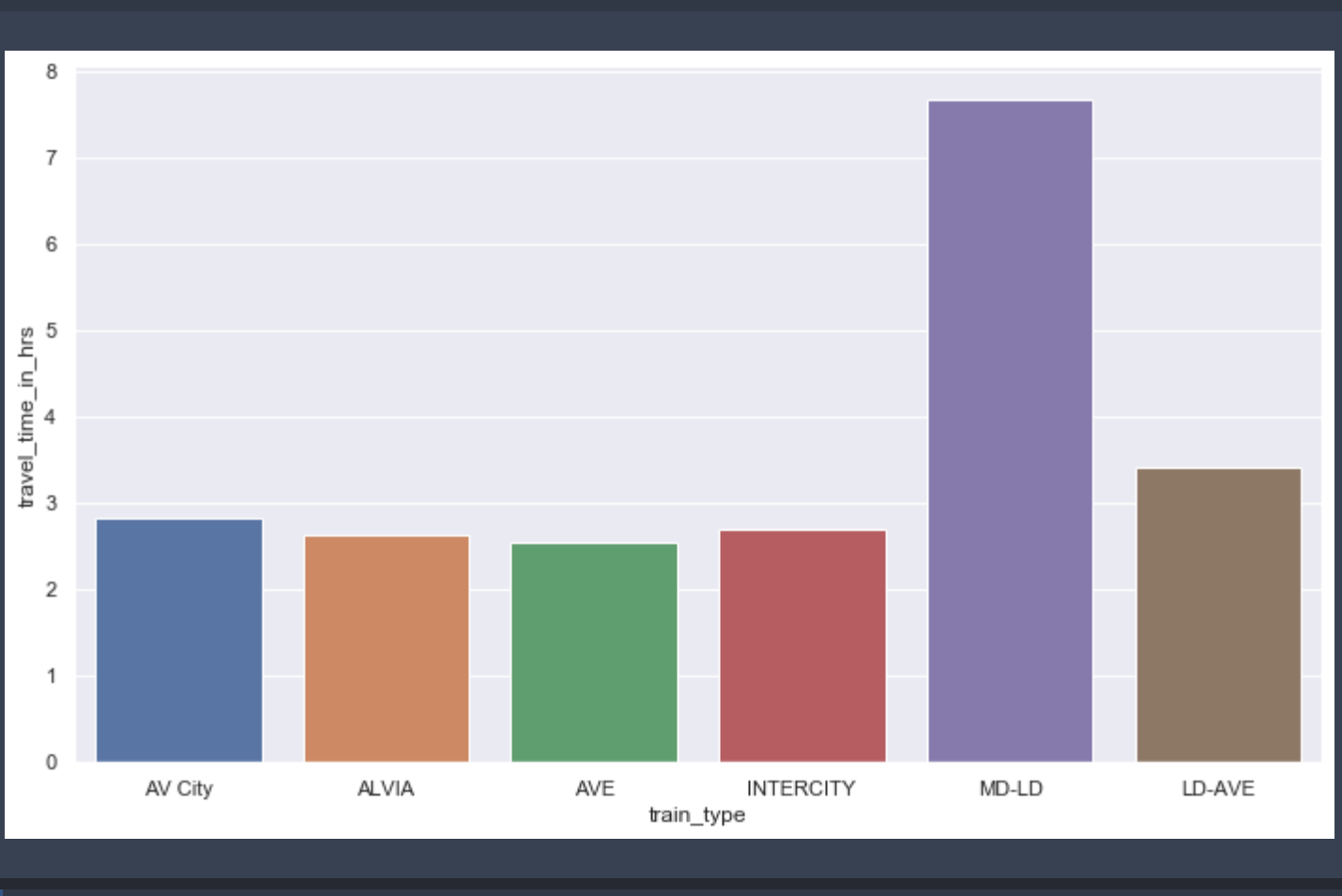
Task 21: Findout people travelling from MADRID to SEVILLA

```
#write code here
df1 = df[(df['origin'] == 'MADRID') & (df['destination'] == 'SEVILLA')]
print("No. of people travelling from MADRID to SEVILLA = "+str(df1.shape[0]))
```

```
No. of people travelling from MADRID to SEVILLA = 26961
```

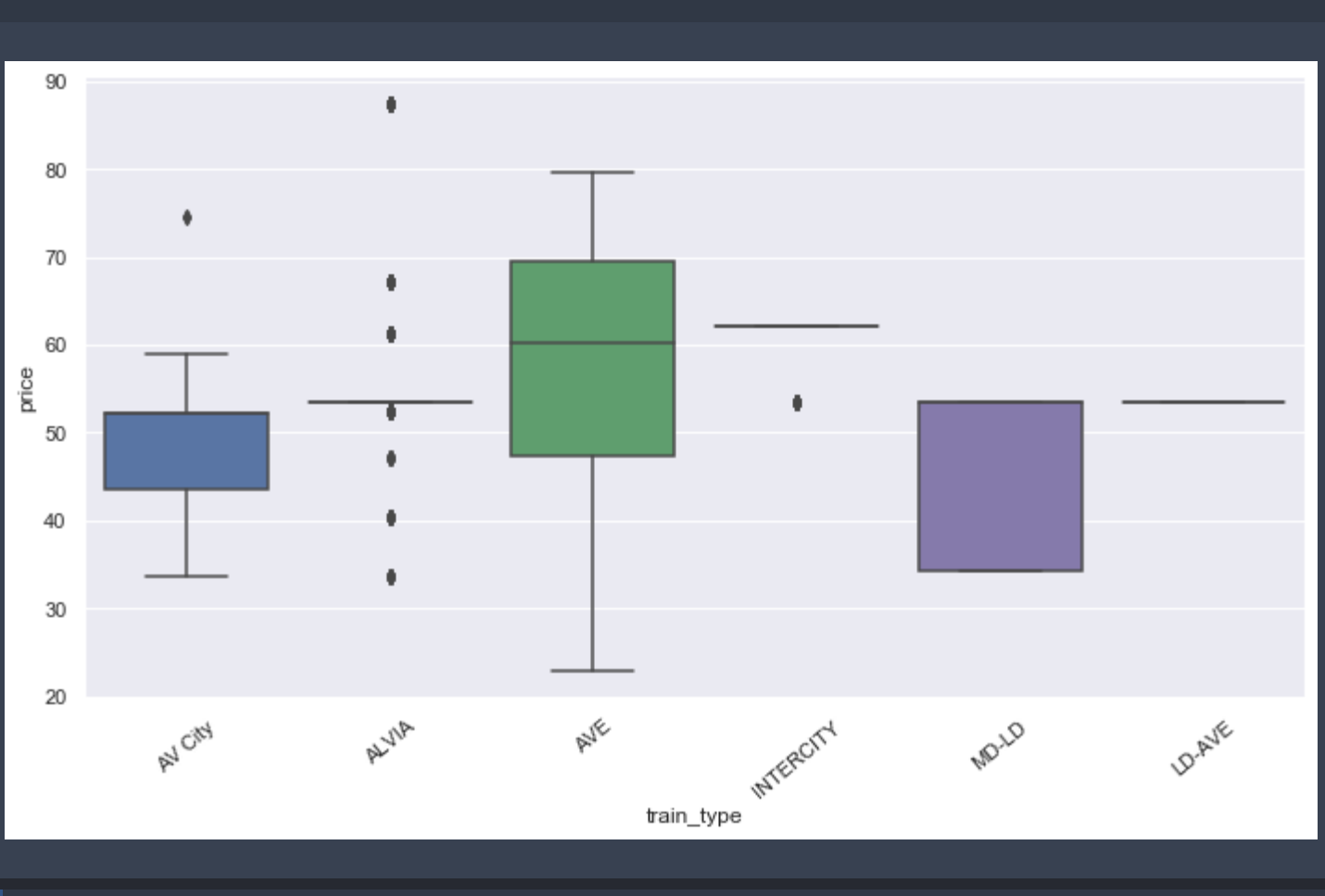
Task 22: Make a plot for finding out travelling hours for each train type

```
#write code here
plt.figure(figsize=(10,6))
ax = sns.barplot(x="train_type", y="travel_time_in_hrs", data=df1, ci = 0)
ax.set_xticklabels(ax.get_xticklabels(), rotation=0, ha="center")
plt.tight_layout()
plt.show()
```



Task 23: Show train_type vs price through boxplot

```
#write code here
plt.figure(figsize=(10,6))
ax = sns.boxplot(x = 'train_type', y = 'price', data = df1)
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="center")
plt.tight_layout()
plt.show()
```



Travelling from MADRID to BARCELONA

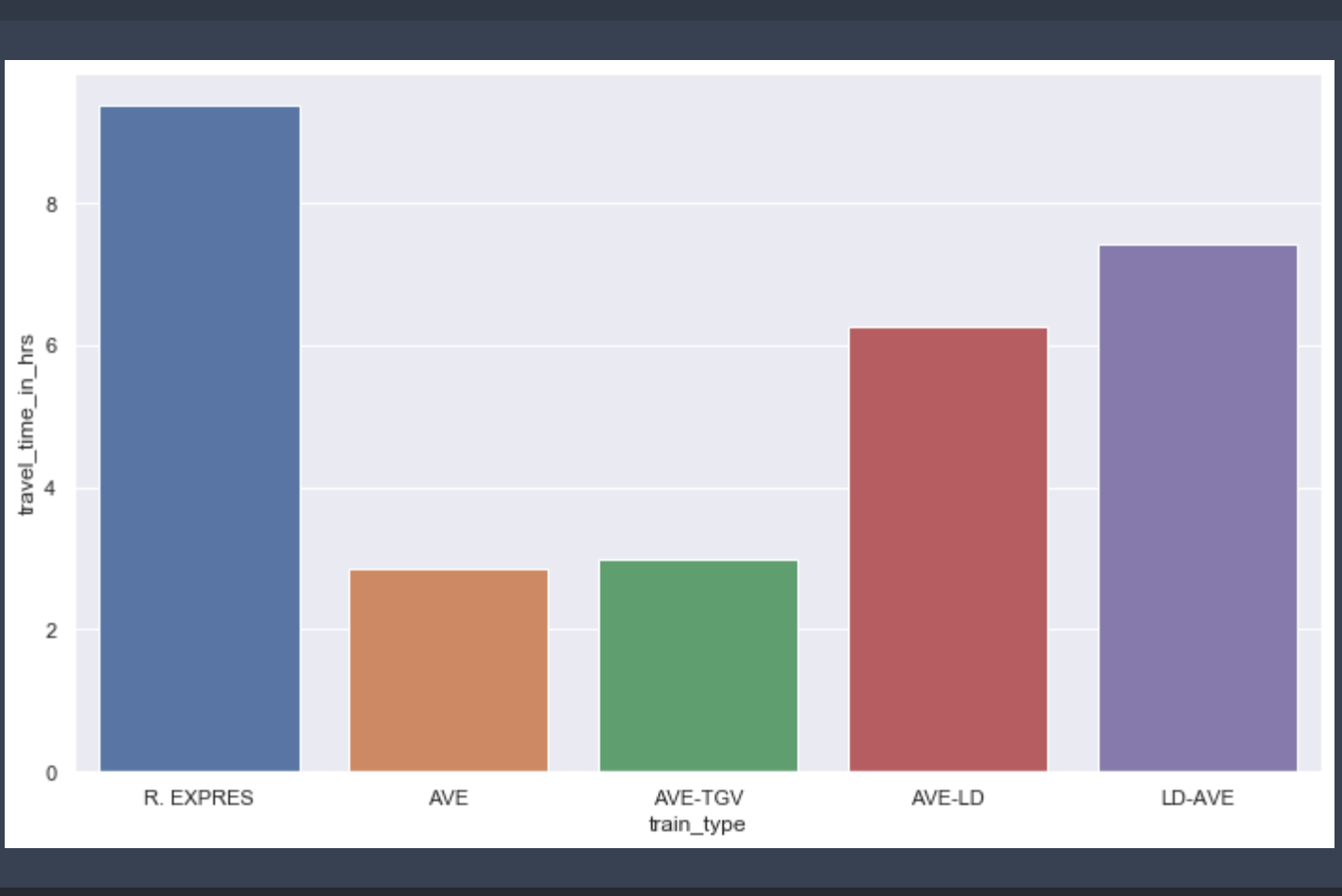
Task 24: Findout people travelling from MADRID to BARCELONA

```
#write code here
df2= df[(df['origin'] == 'MADRID') & (df['destination'] == 'BARCELONA')]
print("No. of people travelling from MADRID to BARCELONA = "+str(df2.shape[0]))
```

```
No. of people travelling from MADRID to BARCELONA = 40728
```

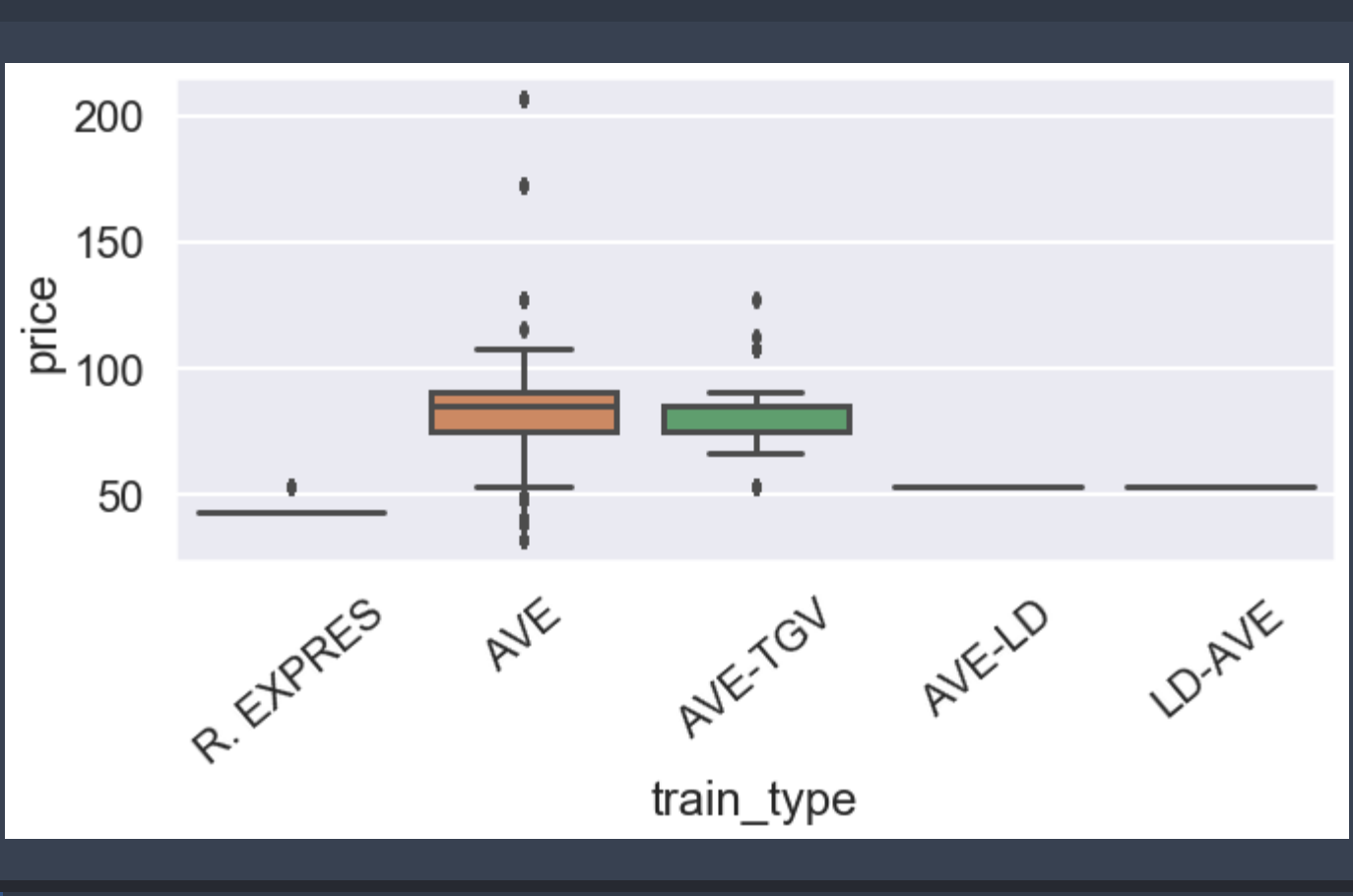
Task 25: Make a plot for finding out travelling hours for each train type

```
#write code here
plt.figure(figsize=(10,6))
ax = sns.barplot(x="train_type", y="travel_time_in_hrs", data=df2, ci = 0)
ax.set_xticklabels(ax.get_xticklabels(), rotation=0, ha="center")
plt.tight_layout()
plt.show()
```



Task 26: Show train_type vs price through boxplot

```
#write code here
plt.figure(figsize=(10,6))
ax = sns.boxplot(x = 'train_type', y = 'price', data = df1)
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="center")
plt.tight_layout()
plt.show()
```



Travelling from MADRID to VALENCIA

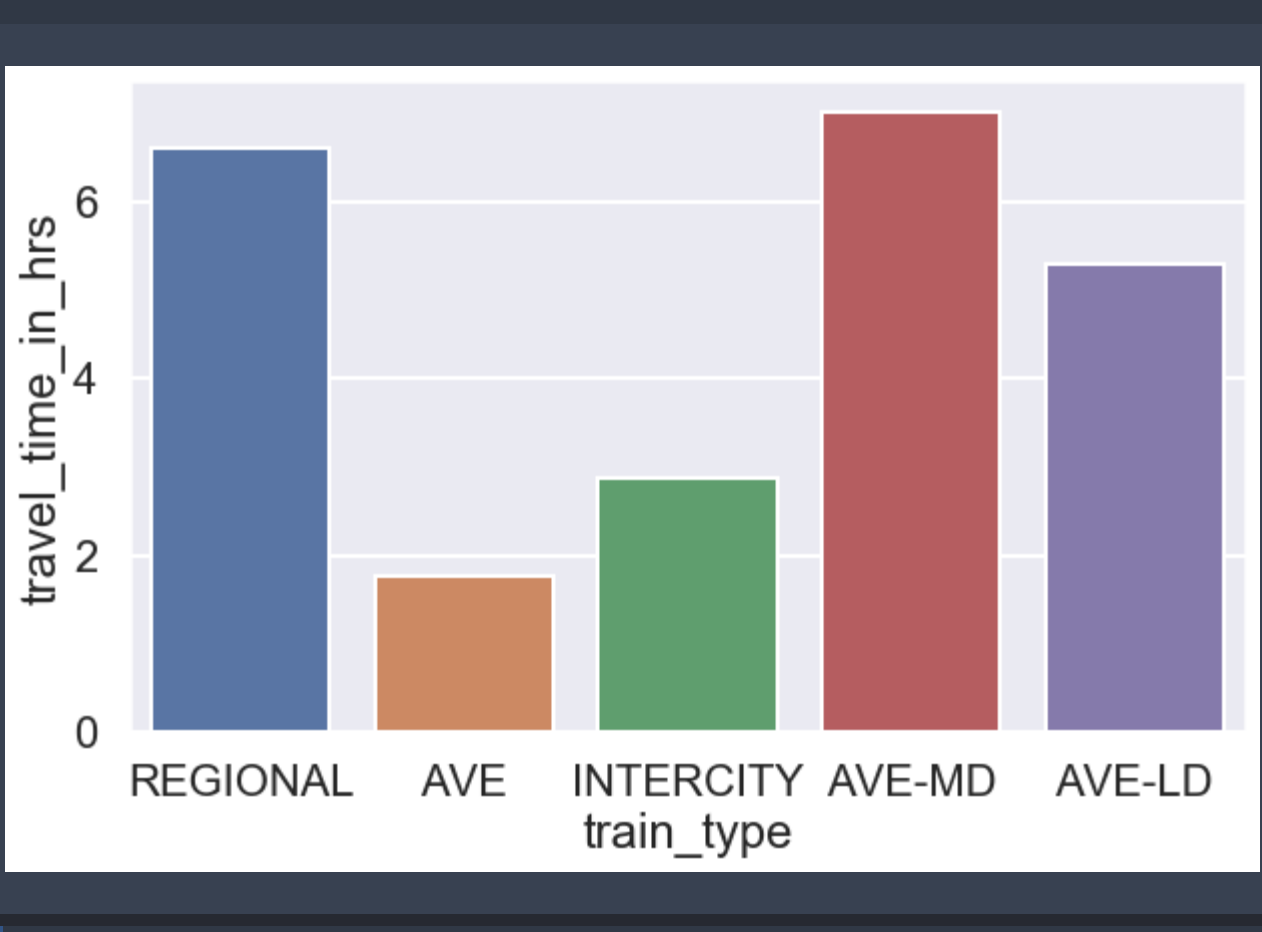
Task 27: Findout people travelling from MADRID to VALENCIA

```
#write code here
df3 = df[(df['origin'] == 'MADRID') & (df['destination'] == 'VALENCIA')]
print("No. of people travelling from MADRID to VALENCIA = "+str(df3.shape[0]))
```

```
No. of people travelling from MADRID to VALENCIA = 39447
```

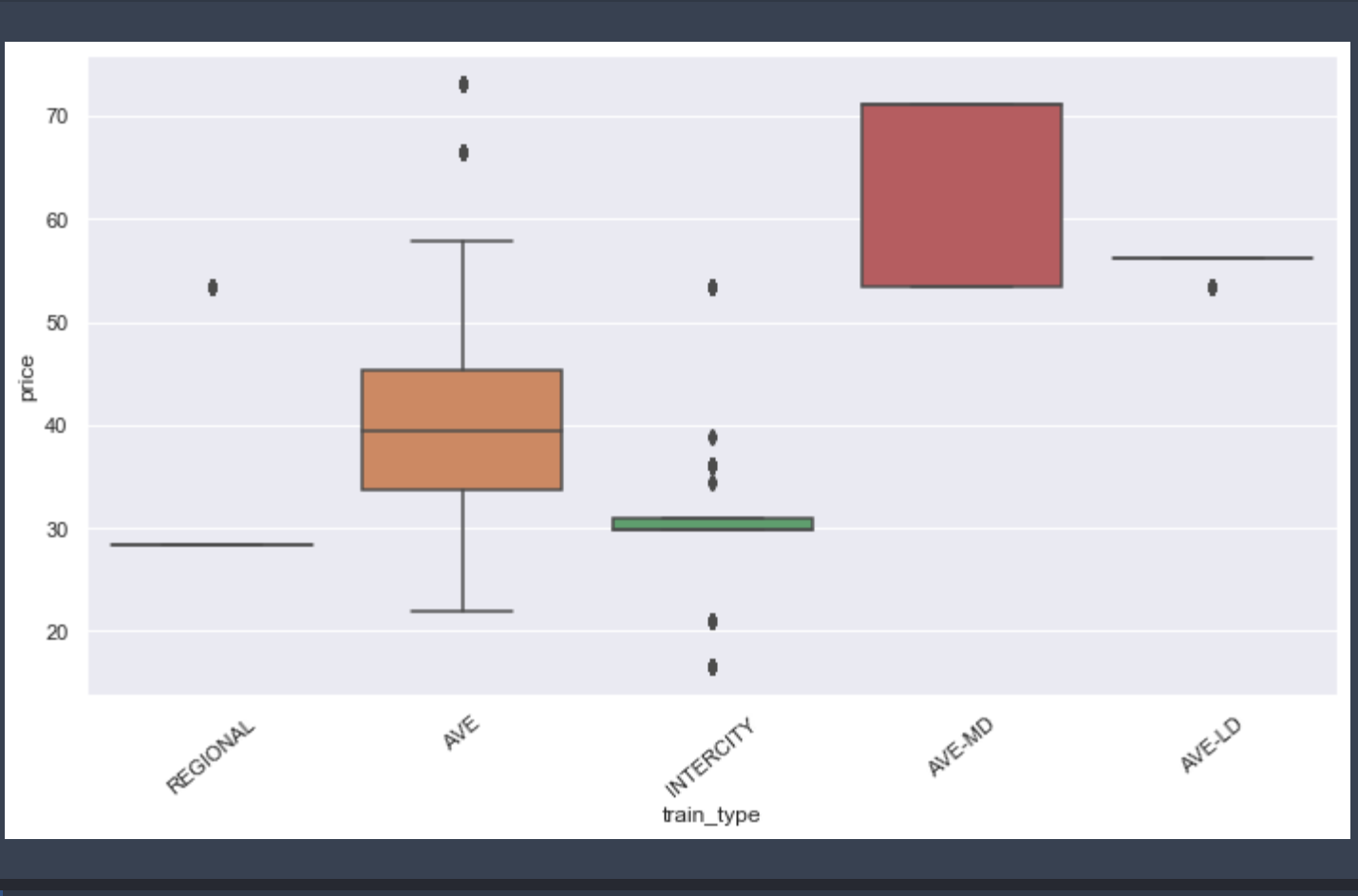
Task 28: Make a plot for finding out travelling hours for each train type

```
#write code here
plt.figure(figsize=(10,6))
ax = sns.barplot(x="train_type", y="travel_time_in_hrs", data=df3, ci = 0)
ax.set_xticklabels(ax.get_xticklabels(), rotation=0, ha="center")
# plt.tight_layout()
plt.show()
```



Task 29: Show train_type vs price through boxplot

```
#write code here
plt.figure(figsize=(10,6))
ax = sns.boxplot(x = 'train_type', y = 'price', data = df3)
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="center")
plt.tight_layout()
plt.show()
```



Travelling from MADRID to PONFERRADA

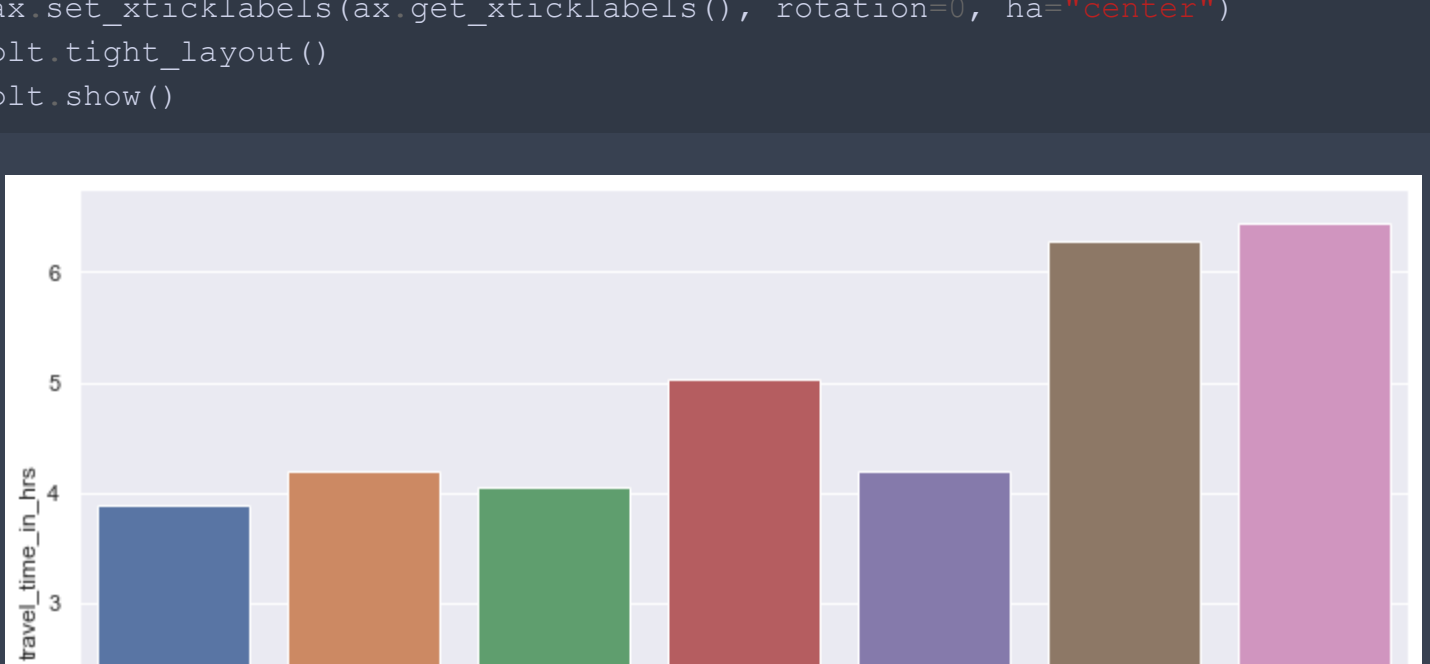
Task 30: Findout people travelling from MADRID to PONFERRADA

```
#write code here
df4 = df[(df['origin'] == 'MADRID') & (df['destination'] == 'PONFERRADA')]
print("No. of people travelling from MADRID to PONFERRADA = "+str(df4.shape[0]))
```

```
No. of people travelling from MADRID to PONFERRADA = 2839
```

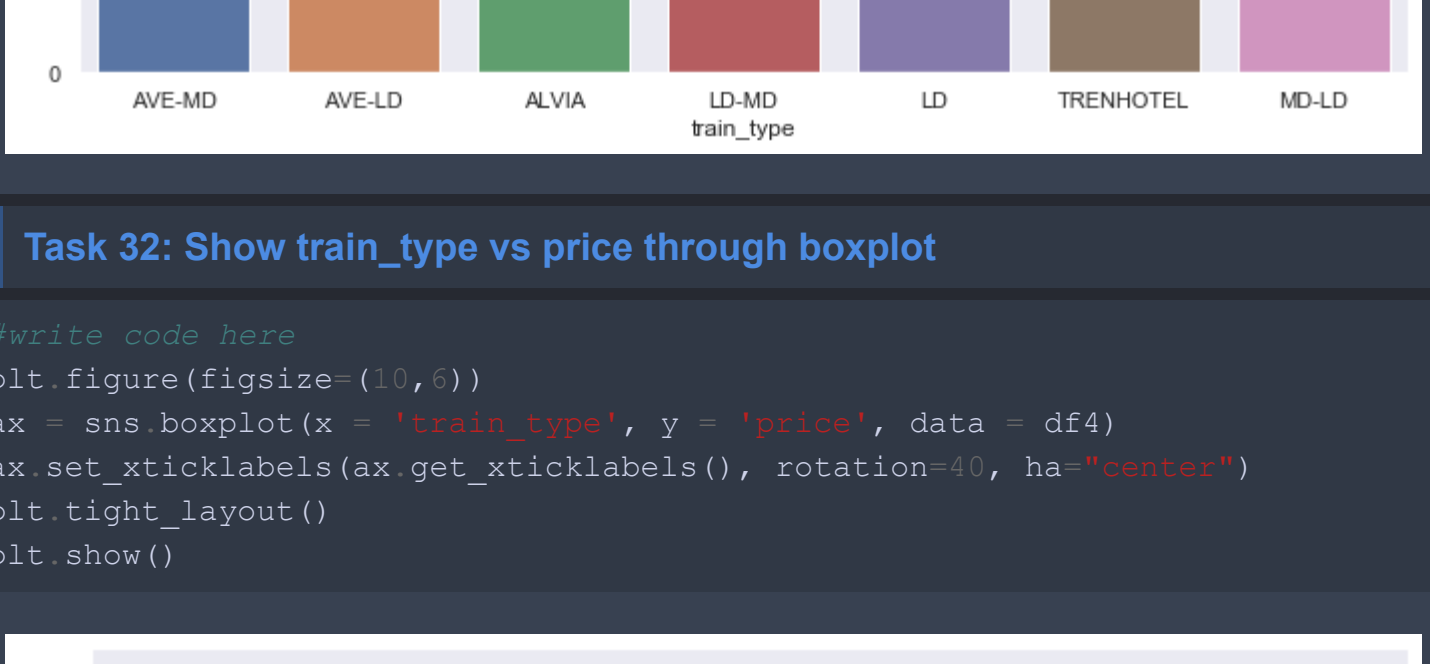
Task 31: Make a plot for finding out travelling hours for each train type

```python
#write code here
plt.figure(figsize=(10,6))
ax = sns.barplot(x="train_type", y="travel_time_in_hrs", data=df4, ci = 0)
ax.set_xticklabels(ax.get_xticklabels(), rotation=0, ha="center")
plt.tight_layout()
plt.show()
```



Task 32: Show train_type vs price through boxplot

```python
#write code here
plt.figure(figsize=(10,6))
ax = sns.boxplot(x = 'train_type', y = 'price', data = df4)
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="center")
plt.tight_layout()
plt.show()
```



Applying Linear Regression

Task 33: Import LabelEncoder library from sklearn

```python
#write code here
from sklearn.preprocessing import LabelEncoder
```

Data Encoding

```python
lab_en = LabelEncoder()
df.iloc[:,1] = lab_en.fit_transform(df.iloc[:,1])
df.iloc[:,2] = lab_en.fit_transform(df.iloc[:,2])
df.iloc[:,3] = lab_en.fit_transform(df.iloc[:,3])
df.iloc[:,5] = lab_en.fit_transform(df.iloc[:,5])
df.iloc[:,6] = lab_en.fit_transform(df.iloc[:,6])
```

```python
df.head()
```

	index	origin	destination	train_type	price	train_class	fare	travel_time_in_hrs
0	0	2	1	11	59.50	4	1	4.700000
1	1	2	1	11	34.65	4	4	5.800000
2	2	2	1	12	39.95	4	4	5.916667
3	3	2	1	11	40.60	4	4	4.983333
4	4	2	1	0	27.90	2	3	4.133333

Task 34: Separate the dependant and independant variables

```python
#write code here
X = df.drop(['price'], axis=1)
Y = df[['price']]
print(X.shape)
print(Y.shape)
```

```
(215266, 7)
(215266, 1)
```

Task 35: Import test_train_split from sklearn

```python
#write code here
from sklearn.model_selection import train_test_split
```

Task 36:**Split the data into training and test set**

```python
#write code here
X_train,X_test,Y_train,Y_test = train_test_split(X, Y, test_size=0.30, random_state=25,shuf
fle=True)
```

Task 37: Import LinearRegression library from sklearn

```python
#write code here
from sklearn.linear_model import LinearRegression
```

Task 38: Make an object of LinearRegression( ) and train it using the training data set

```python
#write code here
lr =  LinearRegression()
```

```python
#write code here
lr.fit(X_train, Y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Task 39: Find out the predictions using test data set.

```python
#write code here
lr_predict = lr.predict(X_test)
```

Task 40: Find out the predictions using training data set.

```python
#write code here
lr_predict_train = lr.predict(X_train)
```

Task 41: Import r2_score library form sklearn

```python
#write code here
from sklearn.metrics import r2_score
```

Task 42: Find out the R2 Score for test data and print it.

```python
#write code here
lr_r2_test= r2_score(Y_test,lr_predict)
print("r2 score of testing "+ str(lr_r2_test))
```

```
r2 score of testing 0.7290143663235672
```

Task 43: Find out the R2 Score for training data and print it.

```python
lr_r2_train =r2_score(Y_train,lr_predict_train)
print("r2 score of training "+ str(lr_r2_train))
```

```
r2 score of training 0.7238620495006879
```

Comaparing training and testing R2 scores

```python
print('R2 score for Linear Regression Training Data is: ', lr_r2_train)
print('R2 score for Linear Regression Testing Data is: ', lr_r2_test)
```

```
R2 score for Linear Regression Training Data is:  0.7238620495006879
R2 score for Linear Regression Testing Data is:  0.7290143663235672
```

Applying Polynomial Regression

Task 44: Import PolynomialFeatures from sklearn

```python
#write code here
from sklearn.preprocessing import PolynomialFeatures
```

Task 45: Make and object of default Polynomial Features

```python
#write code here
poly_reg = PolynomialFeatures(degree=2)
```

Task 46: Transform the features to higher degree features.

```python
#write code here
X_train_poly,X_test_poly = poly_reg.fit_transform(X_train),poly_reg.fit_transform(X_test)
```

```python
print(X_train.shape)
print(X_train_poly.shape)
print(X_test_poly.shape)
```

```
(150686, 7)
(150686, 36)
(64580, 36)
```

Task 47: Fit the transformed features to Linear Regression

```python
#write code here
poly_model = LinearRegression()
poly_model.fit(X_train_poly, Y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Task 48: Find the predictions on the data set

```python
#write code here
y_train_predicted,y_test_predict = poly_model.predict(X_train_poly),poly_model.predict(X_te
st_poly)
```

Task 49: Evaluate R2 score for training data set

```python
#evaluating the model on training dataset
#write code here
r2_train =r2_score(Y_train, y_train_predicted)
```

Task 50: Evaluate R2 score for test data set

```python
# evaluating the model on test dataset
#write code here
r2_test = r2_score(Y_test, y_test_predict)
```

Comaparing training and testing R2 scores

```python
#write code here
print ('The r2 score for training set is: ',r2_train)
print ('The r2 score for testing set is: ',r2_test)
```

```
The r2 score for training set is:  0.819004180163414
The r2 score for testing set is:  0.8195478271895499
```

Task 51: Select the best model

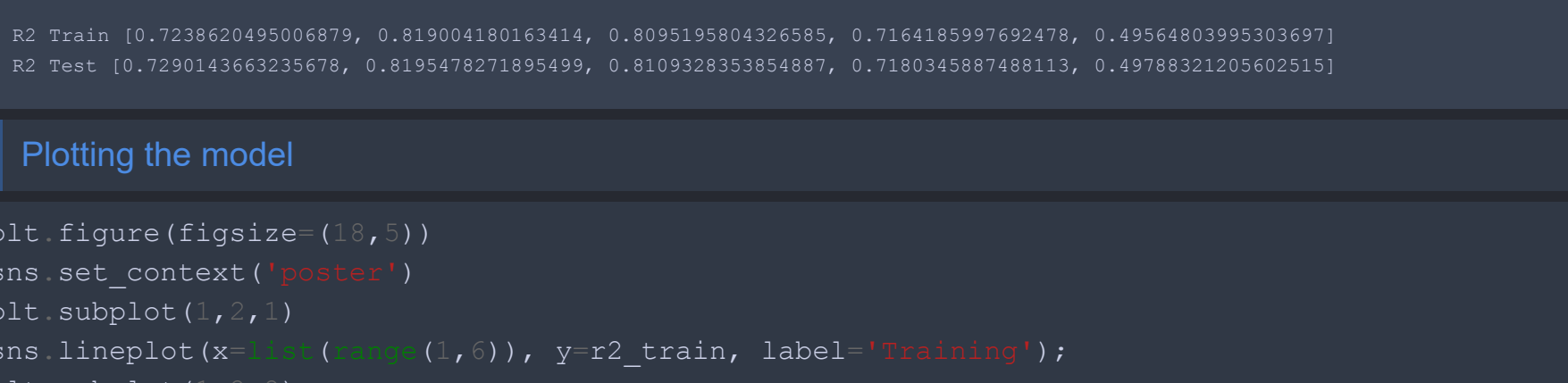
Question: Which model gives the best result for price prediction? Find out the complexity using R2 score and give your answer.
Hint: Use for loop for finding the best degree and model complexity for polynomial regression model

```python
#write code here
r2_train=[]
r2_test=[]
for i in range(1,6):
    poly_reg = PolynomialFeatures(degree=i)

    X_tr_poly,X_tst_poly = poly_reg.fit_transform(X_train),poly_reg.fit_transform(X_test)
    poly =LinearRegression()
    poly.fit(X_tr_poly, Y_train)

    y_tr_predicted,y_tst_predict = poly.predict(X_tr_poly),poly.predict(X_tst_poly)
    r2_train.append(r2_score(Y_train, y_tr_predicted))
    r2_test.append(r2_score(Y_test, y_tst_predict))

print ('R2 Train', r2_train)
print ('R2 Test', r2_test)
```

```
R2 Train [0.7238620495006879, 0.819004180163414, 0.8095195804326585, 0.7164185997692478, 0.49564803995303697]
R2 Test [0.7290143663235678, 0.8195478271895499, 0.8109328353854887, 0.7180345887488113, 0.49788321205602515]
```

Plotting the model

```python
plt.figure(figsize=(18,5))
sns.set_context('poster')
plt.subplot(1,2,1)
sns.lineplot(x=list(range(1,6)), y=r2_train, label='Training');
plt.subplot(1,2,2)
sns.lineplot(x=list(range(1,6)), y=r2_test, label='Testing');
```



Answer

Model with Polynomial degree 2 give the best optimum solution with accuracy of 81% and r2 score 0.81 as well as for training and testing data.