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

در شروع کار تعدادی از کتابخانه هایی را که در ادامه به آنها نیاز خواهیم داشت را فراخوانی میکنیم

Pandas : برای خواندن و استفاده کردن از فایل csv

Numpy : چون با ماتریسها کار میکنیم، در بعضی جاها به آن نیاز خواهیم داشت

Seaborn & matplotlib : برای رسم نمودارها استفاده میکنیم.

```
#Read Data
Data1=pd.read_csv("D:/M.B 4002/Deep Learning- DR Ghaderi/Exer2/Q1/House Sales.csv")
```

با این دستور آدرسی که فایلمون رو در آنجا ذخیره کردیم را وارد میکنیم تا فایلمون خوانده بشه.

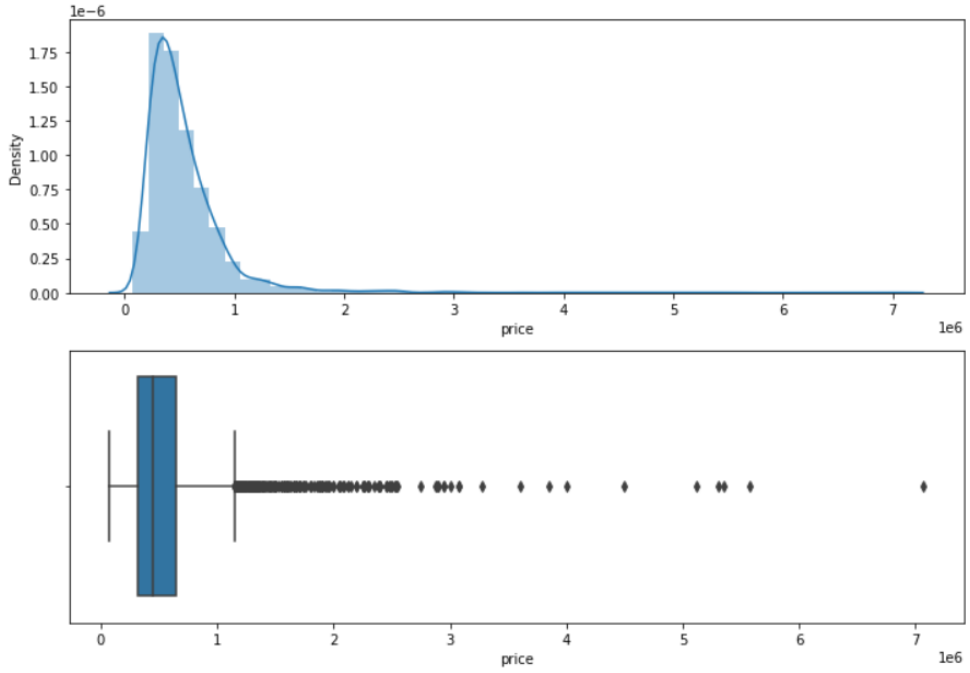
```
#Take first 5000 data
Data=Data1[:5000]

#Get informatin of Data
Data.info()
Data.describe().transpose()
```

روی سوال از ما خواسته ۵۰۰۰ دیتا اول را جدا کرده و با آنها کار کنیم. ۵۰۰۰دیتای اول را جدا کرده و با دستورات info و describe دیتا را نگاه کلی می کنیم و اطلاعات کلی راجع به آن را به دست می آوریم.

```
#Visualizing house prices
fig = plt.figure(figsize=(10,7))
fig.add_subplot(2,1,1)
sns.distplot(Data['price'])
fig.add_subplot(2,1,2)
sns.boxplot(Data['price'])
plt.tight_layout()
```

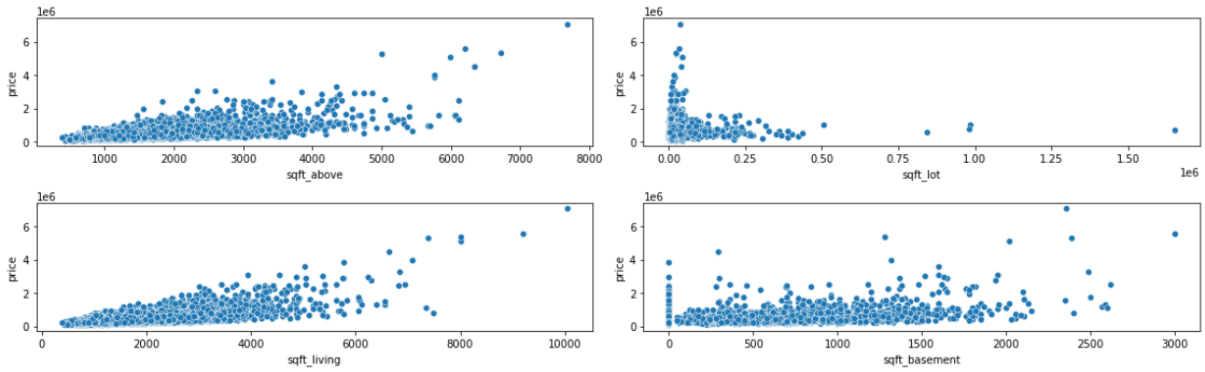
هدف ما در این سوال پیش‌بینی قیمت خانه است. بر اساس داده هایی که داریم قیمت خانه‌ها را به صورت نمودار درمیآوریم تا تحلیل کنیم.



همانطور که مشاهده میکنیم، توزیع قیمت خانه بیشتر در بازه ۰/۵ تا ۱ میلیون دلار است. تعداد خیلی کمی هم در نزدیکی‌های ۶ تا ۷ میلیون دلار داریم که در اینجا برای ما outlier محسوب میشوند.

```
#Visualizing square footage of (home,lot,above and basement)
fig = plt.figure(figsize=(16,5))
fig.add_subplot(2,2,1)
sns.scatterplot(Data['sqft_above'], Data['price'])
fig.add_subplot(2,2,2)
sns.scatterplot(Data['sqft_lot'],Data['price'])
fig.add_subplot(2,2,3)
sns.scatterplot(Data['sqft_living'],Data['price'])
fig.add_subplot(2,2,4)
sns.scatterplot(Data['sqft_basement'],Data['price'])
plt.tight_layout()
```

در ادامه برای اینکه تاثیر پارامتر های مختلف را روی قیمت خانه بتوانیم بررسی کنیم. Scatterplotهای چندین ویژگی را نسبت به قیمت خانه رسم میکنیم که نتایج ان در زیر قابل مشاهده است.

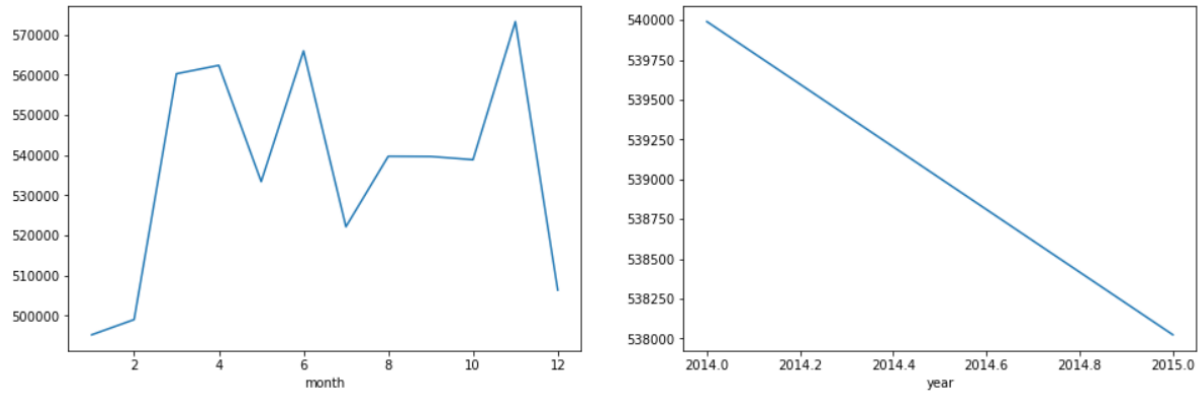


```
#Break date to years, months
Data['date'] = pd.to_datetime(Data['date'])
Data['month'] = Data['date'].apply(lambda date:date.month)
Data['year'] = Data['date'].apply(lambda date:date.year)
```

مورد بعدی که میخواهیم بررسی کنیم. روند تغییرات قیمت خانه در طی سال و ماه‌های مختلف است. برای این منظور از دیتا اصلی با دستور to_datetime ستون مربوط به تاریخ را برداشته و در ادامه با دستور apply و به کمک تابع lambda سال و ماه که در کنار هم نوشته شده را تفکیک می‌کنیم.

```
#Data visualization house price vs months and years
fig = plt.figure(figsize=(16,5))
fig.add_subplot(1,2,1)
Data.groupby('month').mean()['price'].plot()
fig.add_subplot(1,2,2)
Data.groupby('year').mean()['price'].plot()
```

سپس، برای تحلیل بهتر نمودار رسم می‌کنیم به این صورت که محور افقی نشان دهنده‌ی ماه یا سال و محور عمودی نشان دهنده‌ی میانگین قیمت خانه‌هایی است که در آن ماه یا سال خریداری شده است که با استفاده از دستور groupby().mean() به دست می‌آید.



نتایج حاصل نشان میدهد که قیمت خانه در طی ماه ۲ تا ۱۲ در هرماه نوسانات زیادی داشته ولی در طی یک سال از ۲۰۱۴ تا ۲۰۱۵ به حالت کلی سیر نزولی داشته است.

```
#Check if there are any Null values
Data.isnull().sum()
#Drop some unnecessary columns
Data = Data.drop('id',axis=1)
Data = Data.drop('zipcode',axis=1)
```

پس از انجام یه سری تحلیل‌ها و اینکه آشنایی نسبی با دیتاهامون پیدا کردیم، دیتا را برای مدلسازی آماده می-کنیم . یعنی پیش پردازش انجام میدهیم.

در ابتدا با دستور isnull() بررسی می کنیم که آیا در دیتاست ما ستونی وجود دارد که مقدار نداشته باشد یا به اصطلاح خالی باشد؟ دیتاست ما ستون خالی ندارد.

در ادامه با دستور drop ویژگی‌هایی که تو دیتاست داریم ولی تو پیش‌بینی قیمت خانه تاثیر ندارند، مثل کدپستی یا id رو حذف می‌کنیم.

```
X = Data.drop('price',axis =1).values
y = Data['price'].values
```

در مرحله بعد target valueرا که در این سوال همان priceیا قیمت خانه است را از دیتاست جدا کرده و ورودی و خروجی ها را مشخص می‌کنیم.

با دستور dropستون قیمت را حذف کرده و مقادیر مابقی ستون‌ها را به عنوان featureهای ورودی می-گیریم و متغیر X نام‌گذاری می‌کنیم.

در متغیر y که همان خروجی یا outputها هستند فقط مقادیر ستون price را میریزیم.

```
#Splitting Train and Test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8)
```

روی سوال از ما خواسته که ۸۰٪ داده‌ها را آموزش و ۲۰٪ را تست در نظر بگیریم. در این مرحله این تقسیم بندی را با استفاده از تابع train_test_split که از کتابخانه sklearnفراخوانی کردیم انجام میدهیم. و Train_size را ۰/۸ مینویسیم یعنی ۸۰٪ داده ها برای train باشد.

```
#Standardization scaler - fit&transform on train, fit only on test
from sklearn.preprocessing import StandardScaler
s_scaler = StandardScaler()
X_train = s_scaler.fit_transform(X_train.astype(np.float64))
X_test = s_scaler.transform(X_test.astype(np.float64))
```

در مرحله بعد داده هارا استانداردسازی می‌کنیم. با استفاده از تابع StandardScalerاز کتابخانه sklearn .

یعنی روی داده های آموزش و تست میانگین را remove کرده و به داده‌ها یه واریانس واحد scale می‌کنیم.

```
#Creating a Neural Network Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam

# having 19 neuron is based on the number of available features with 1 hidden layer
model = Sequential()
model.add(Dense(19,activation='relu'))
model.add(Dense(1))
model.compile(optimizer='Adam',loss='MSE')
```

حال نوبت مدلسازی هست. مدل را sequentialانتخاب کردیم و آن را از کتابخانه tensorflowفراخوانی کردیم. و همچنین برای ساختن لایه‌های مخفی به dense و activation function نیاز داریم و برای compileکردن مدل از optimizer ، Adam میخواهیم استفاده کنیم که همه ان هارا از tensorflow فراخوانی کردیم.

بر اساس اطلاعاتی که در ابتدای کد ار دیتاست به دست اوردیم دیدیم که ۲۱ تا featureداریم و با حذف کردن ۲تا از آنها (کدپستی و id) ۱۹ تا ویژگی موند که تعداد نرون‌های لایه مخفی اول را همون ۱۹ تا در نظر میگیریم. و تابع فعالسازی relu رو هم میدیم بهش. و چون یک لایه مخفی میخوایم داشته باشیم لایه بعدی همان لایه خروجی هست و چون فقط یه خروجی خواهیم داشت که قیمت خانه است پس تعداد نرون لایه خروجی ۱ خواهد بود. و در نهایت با optimizer adam و تابع loss که اینجا از MSE(Mean Square Error) استفاده کردیم ، مدلمون رو کامپایل می‌کنیم.

```python
#Evaluation
y_pred = model.predict(X_test)
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print('VarScore:',metrics.explained_variance_score(y_test,y_pred))

# Visualizing Our predictions
fig = plt.figure(figsize=(10,5))
plt.scatter(y_test,y_pred)

# Perfect predictions
plt.plot(y_test,y_test,'r')
plt.tight_layout()

#Visualizing residuals
fig = plt.figure(figsize=(10,5))
residuals = (y_test - y_pred)
sns.distplot(residuals)
```

در مرحله آخر مدلی که ساختیم رو روی داده های تست ارزیابی می‌کنیم با دستور predict()

برای بدست آوردن خطاهای مدلمون (MAE, MSE, ..) از تابع Metricsدر sklearn استفاده می‌کنیم . از مقادیر آنها پرینت میگیریم.

در ادامه با نشان دادن مقادیر yپیش‌بینی شده توسط مدل و مقادیرy واقعی روی نمودار بررسی میکنیم که مدلی که ساختیم تا چه حد درست کار میکند.

و در مرحله آخر هم خطای بین مقادیر واقعی و پیش بینی شده رو تحت عنوان residualsنمایش میدهیم.

```
MAE: 536952.6346266053
MSE: 467422450564.70465
RMSE: 683683.0044433639
VarScore: 1.9001955835484807e-06
```

اعدادی که برای مدل MLPبا یک لایه مخفی به دست آمد به این صورت بود.

مقدار varscore در بهترین حالت مقدار ۱ را دارد.

```
# having 19 neuron is based on the number of available features with 2 hidden layer
model = Sequential()
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(1))
model.compile(optimizer='Adam',loss='MSE')
```

مدل دو لایه مخفی و در هر لایه ۱۹ نرون

```
MAE: 536952.4826938066
MSE: 467422686111.5926
RMSE: 683683.176706574
VarScore: -3.2592349108462315e-07
```

نتایج:

```
# having 19 neuron is based on the number of available features with 2 hidden layer
model = Sequential()
model.add(Dense(15,activation='relu'))
model.add(Dense(8,activation='relu'))
model.add(Dense(1))
model.compile(optimizer='Adam',loss='MSE')
```

مدل با دو لایه مخفی و لایه اول ۱۵ نرون و لایه دوم ۸ نرون

```
MAE: 536952.6075043398
MSE: 467422727028.4704
RMSE: 683683.2066304323
VarScore: 1.939836378195281e-07
```

نتایج: