سوال 4)

دیتا فریم خود را لود میکنیم

```
!gdown 1LjgD0GCoCoi_nIXp4AkRsyvd8CNd1hY1
df = pd.read_csv('/content/data.csv')
df.info()
```

```
#   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           4600 non-null   object
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64
 5   sqft_lot       4600 non-null   int64
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64
 8   view           4600 non-null   int64
 9   condition      4600 non-null   int64
 10  sqft_above     4600 non-null   int64
 11  sqft_basement  4600 non-null   int64
 12  yr_built       4600 non-null   int64
 13  yr_renovated   4600 non-null   int64
 14  street         4600 non-null   object
 15  city           4600 non-null   object
 16  statezip       4600 non-null   object
                                17  country        4600 non-null   object
```

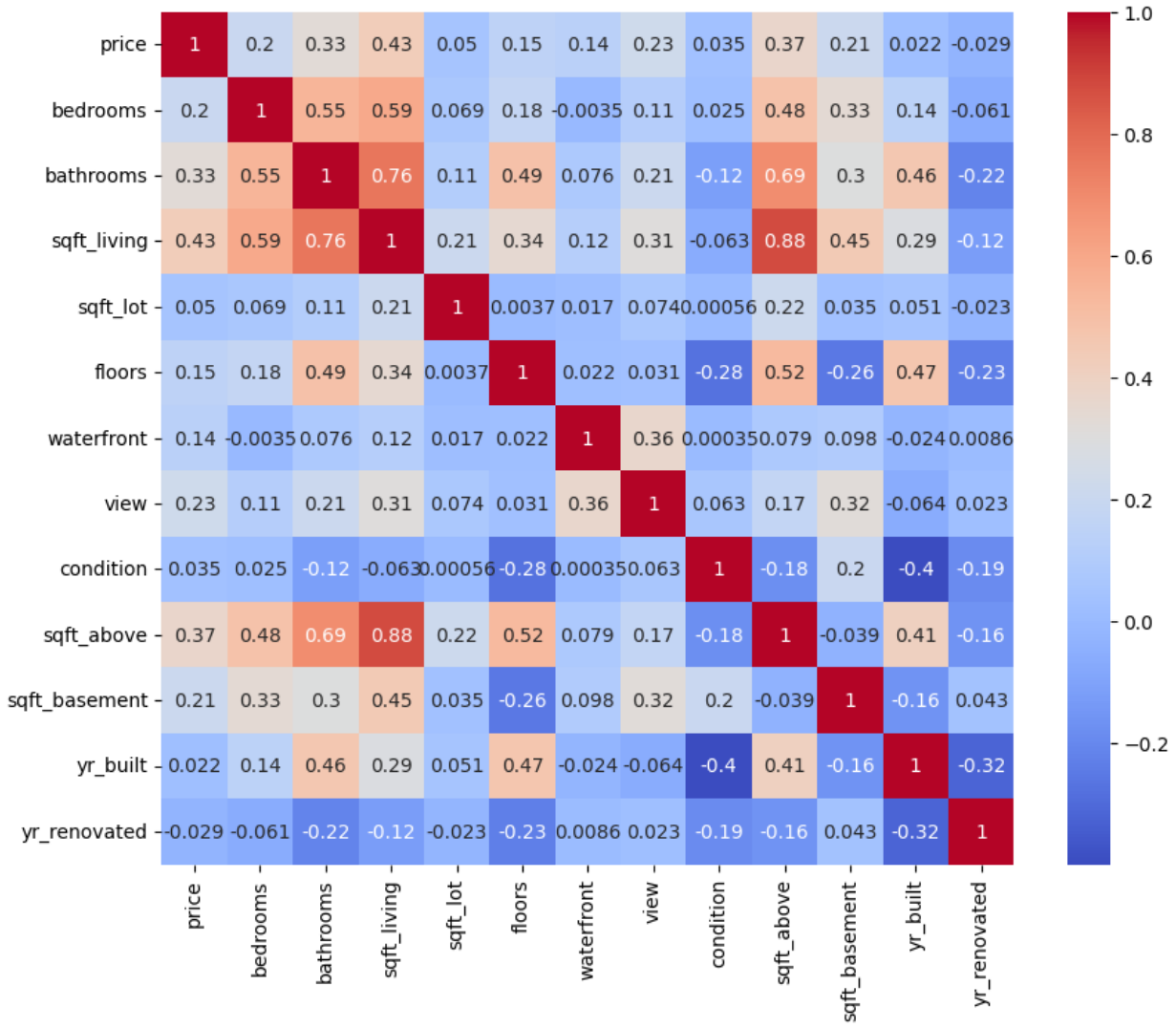
دیتا فریم دارای 17 تا ویژگی و یک قیمت است که میخواهیم براساس این ویژگی ها قیمت را پیشبینی کنیم

```
nan = df.isnull().sum()
nan
#df.dropna(inplace=True)
```

برسی میکنیم که آیا دارای null هست دیتا یا نه که دراین دیتا نداشتیم

```
corr_mat = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_mat, annot=True, cmap="coolwarm", square=True)
plt.show
```

ماتریس کورولیشن را رسم میکنیم تا ببینم چه ویژگی هایی بیش تر روی قیمت اثر دارد

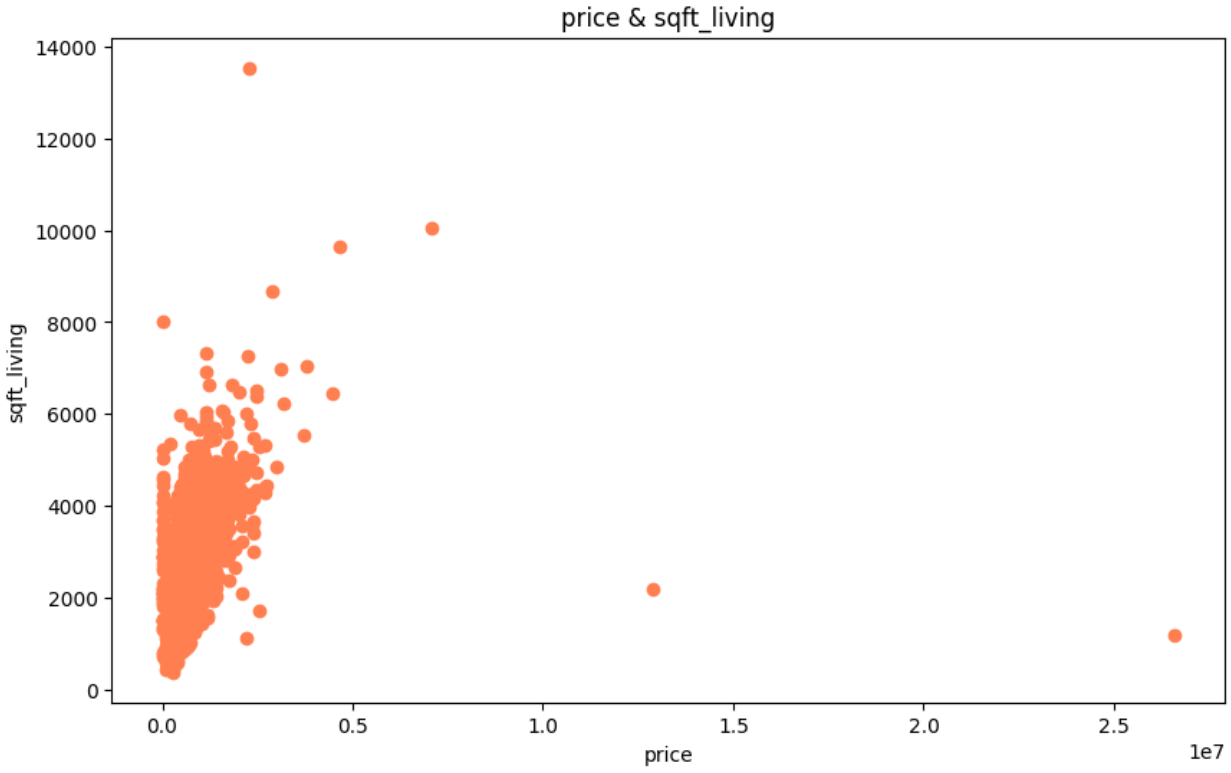


همانطور که مشاهدی میشود sqft_living بیش ترین اثر را روی قیمت خانه دارد

قسمت 3)

```
plt.figure(figsize=(10, 6))
plt.scatter(df['price'], df[max_corr_feature_with_price], color='coral')
plt.title(f'price & {max_corr_feature_with_price}')
plt.xlabel('price')
plt.ylabel(max_corr_feature_with_price)
plt.show()
```

برای رسم این نمودار از scatter.plt ا ستفاده می کنیم و محور x آن را ستون living-sqft ومحورy آن را ستون price قرار می دهیم



قسمت4)

```
df[['year', 'month', 'day']] = df['date'].str.split('-', expand = True)
df = df.drop('date', axis = 1)
df = df.drop('day', axis = 1)
print(df)
```

بدین گونه ستون مربوط به تاریخ را حذف و ستون ماه سال و روز را اضافه میکنیم

قسمت 5)

```
from sklearn.model_selection import train_test_split
X = df.values[:, 1 : 13]
y = df.values[:, 0]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

ابتدا داده های تست و ترین را از هم جدا میکنیم

```
from sklearn.preprocessing import MinMaxScaler
scalerx_train = MinMaxScaler()
scalerx_test = MinMaxScaler()

scalerx_train.fit(X_train)
scalerx_test.fit(X_test)
```

داده های تست و ترین را به طور جداگانه مقایس بندی میکنیم

```
# Normalize the training input data
X_train = scalerx_train.transform(X_train)

# Normalize the test input data
X_test = scalerx_test.transform(X_test)
```

همین کار را برای y انجام میدهیم ولی اول باید shape آن را درست کنیم

```
y_train = y_train[:, None]
y_test = y_test[:, None]
```

```
y_train = scaler_2.fit_transform(y_train)
y_test = scaler_2.transform(y_test)
```

یک شبکه عصبی با دو لایه پنهان درست میکنیم

```
model = Sequential()
model.add(Dense(40, activation= 'relu', input_shape =
(X_train.shape[1],)))
model.add(Dense(20, activation= 'relu'))
model.add(Dense(10, activation= 'relu'))
model.add(Dense(1, activation= 'linear'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 40)                520

 dense_1 (Dense)             (None, 20)                820

 dense_2 (Dense)             (None, 10)                210

 dense_3 (Dense)             (None, 1)                 11
```

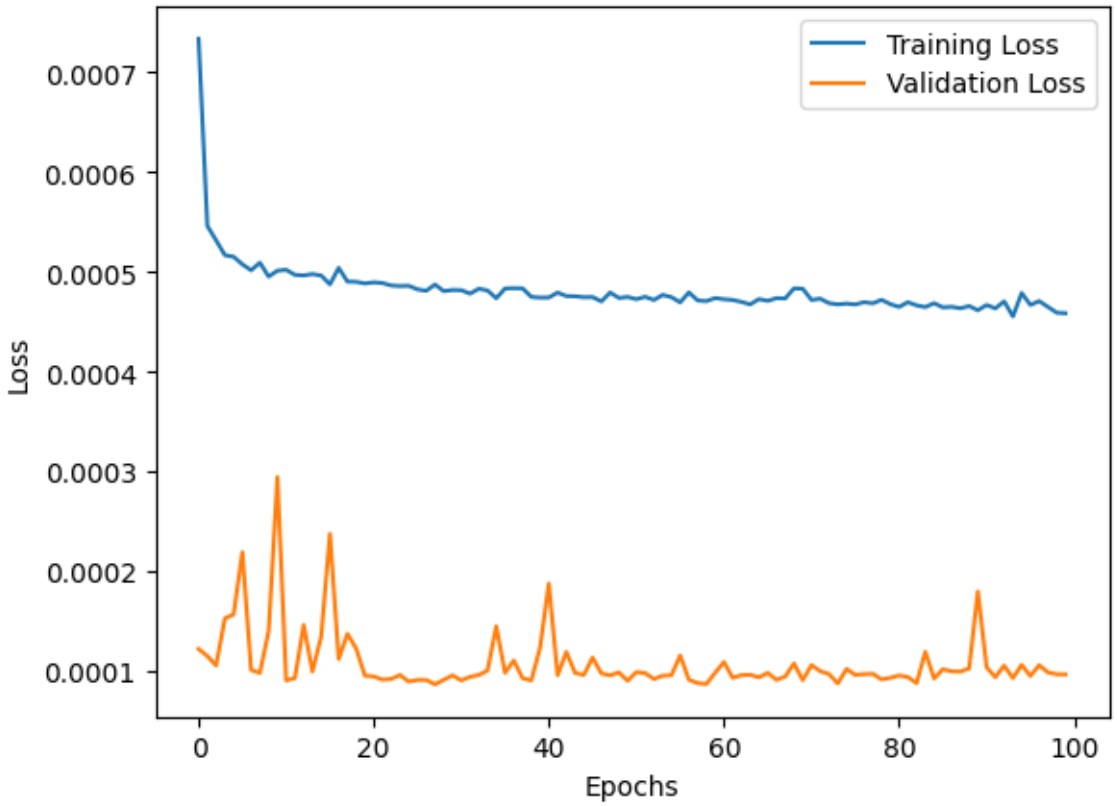
```
model.compile(optimizer='adam', loss='mse')
history = model.fit(X_train, y_train, validation_split=0.2, epochs=100,
batch_size=10)
```

با استفاده از بهینه ساز adam وتابع اتلاف .mse برای آموزش شبکه train-xو train-y را به عنوان ورودی می دهیم و 0.2 داده ها را برای اعتبارسنجی قرار می دهیم.

```
loss = model.evaluate(X_test, y_test)
y_pred_1 = model.predict(X_test)
rscore_1 = r2_score(y_test, y_pred_1)
# Plot the training and validation loss
plt.plot(history.history['loss'], label='train')   # Training loss
plt.plot(history.history['val_loss'], label='val')  # Validation loss

plt.legend(['Training Loss', 'Validation Loss'])
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.show()
```

نمودار را رسم میکنیم



برای رسم نمودار اتلاف آموزش و اعتبار سنجی به این شکل عمل می کنیم.

طبق نتایج بدست آمده، هر چه R۲ score به یک نزدیک تر باشد بهتر است و در اینجا عدد -3.5559106987844853 بدست آمده است و نشان دهنده ی آن است که عملکرد شبکه عصبی ما خوب نمی باشد. برای رفع این مشکل می توانیم لایه های پنهان شبکه عصبی را زیاد تر کنیم و تعداد نرون لایه ها را افزایش دهیم.

همچنین می توانیم با تغییر تابع فعالساز و تابع اتلاف و بهینه ساز و آزمون خطا نتیجه بهتری را کسب کنیم.

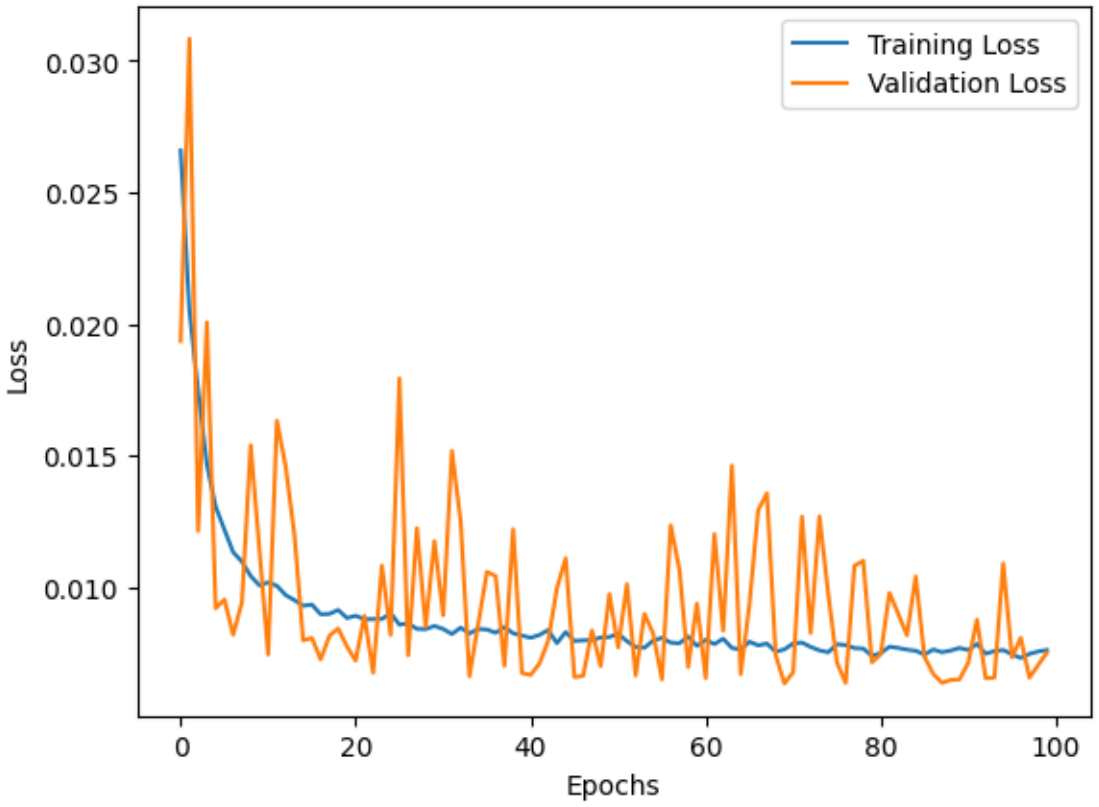
برای حالت دیگر از تابع اتلاف MAE (Error Absolute Mean)و بهینه ساز sgd استفاده می کنیم.

```
model_2.compile(optimizer='sgd', loss='mae')

history = model_2.fit(X_train, y_train, validation_split=0.2, epochs=100,
batch_size=10)
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.legend(['Training Loss', 'Validation Loss'])
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.show()
```



در این حالت R2 عدد -1 به دست آمده و بهبود یافته ولی نمودار اعتبار سنجی تقریبا ناپایدار است که میتوان یکی از دلیل های آن را overfitting است

قسمت آخر)

```
import random

random_pred = list()
random_test = list()

for i in range(5):
  j = random.randint(0, len(y_pred_2) ) # Generate a random index
  random_pred.append(y_pred_2[j]) # Append y_pred_2 value at the
randomindex j
  random_test.append(y_test[j]) # Append y_test value at the same
randomindex j

# Plot the random predictions and actual test outputs
plt.plot(random_pred , 'b', label='Prediction') # Blue line for
predictions
plt.plot(random_test , 'r', label='Test') # Red line for actual
testoutputs

plt.legend()
plt.grid()
plt.show()
```