```
import pandas as pd
import numpy as np
import random as rnd


import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline



from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler


from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam


from sklearn.metrics import mean_squared_error,mean_absolute_error,explained_variance_score
from sklearn.metrics import classification_report,confusion_matrix
```

```
df=pd.read_csv("/content/kc_house_data.csv")
```

```
print(df.columns.values)
```

```
['id' 'date' 'price' 'bedrooms' 'bathrooms' 'sqft_living' 'sqft_lot'
 'floors' 'waterfront' 'view' 'condition' 'grade' 'sqft_above'
 'sqft_basement' 'yr_built' 'yr_renovated' 'zipcode' 'lat' 'long'
 'sqft_living15' 'sqft_lot15']
```

```
df.head()
df.tail()
df.isnull().sum()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
```

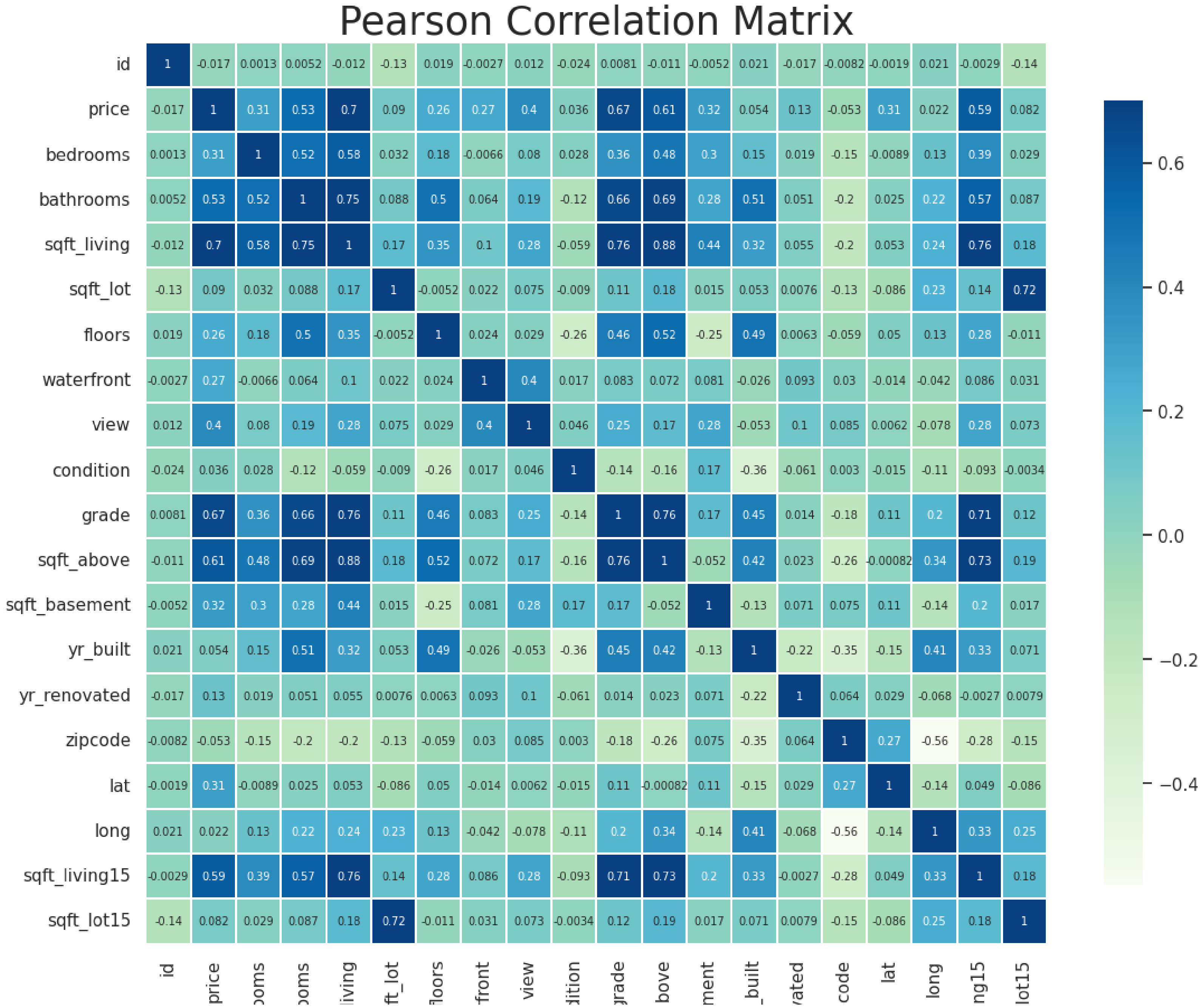
```
 0   id             21613 non-null  int64
 1   date           21613 non-null  object
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64
 6   sqft_lot       21613 non-null  int64
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64
 9   view           21613 non-null  int64
 10  condition      21613 non-null  int64
 11  grade          21613 non-null  int64
 12  sqft_above     21613 non-null  int64
 13  sqft_basement  21613 non-null  int64
 14  yr_built       21613 non-null  int64
 15  yr_renovated   21613 non-null  int64
 16  zipcode        21613 non-null  int64
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64
 20  sqft_lot15     21613 non-null  int64
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

```
df.describe().transpose()
```

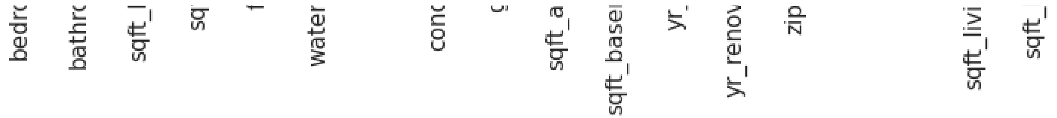
	count	mean	std	min	25%	50%	75%	max
id	21613.0	4.580302e+09	2.876566e+09	1.000102e+06	2.123049e+09	3.904930e+09	7.308900e+09	9.900000e+09
price	21613.0	5.400881e+05	3.671272e+05	7.500000e+04	3.219500e+05	4.500000e+05	6.450000e+05	7.700000e+06
bedrooms	21613.0	3.370842e+00	9.300618e-01	0.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	3.300000e+01
bathrooms	21613.0	2.114757e+00	7.701632e-01	0.000000e+00	1.750000e+00	2.250000e+00	2.500000e+00	8.000000e+00
sqft_living	21613.0	2.079900e+03	9.184409e+02	2.900000e+02	1.427000e+03	1.910000e+03	2.550000e+03	1.354000e+04
sqft_lot	21613.0	1.510697e+04	4.142051e+04	5.200000e+02	5.040000e+03	7.618000e+03	1.068800e+04	1.651359e+06
floors	21613.0	1.494309e+00	5.399889e-01	1.000000e+00	1.000000e+00	1.500000e+00	2.000000e+00	3.500000e+00
waterfront	21613.0	7.541757e-03	8.651720e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
view	21613.0	2.343034e-01	7.663176e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.000000e+00
condition	21613.0	3.409430e+00	6.507430e-01	1.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	5.000000e+00
grade	21613.0	7.656873e+00	1.175459e+00	1.000000e+00	7.000000e+00	7.000000e+00	8.000000e+00	1.300000e+01
sqft_above	21613.0	1.788391e+03	8.280910e+02	2.900000e+02	1.190000e+03	1.560000e+03	2.210000e+03	9.410000e+03
sqft_basement	21613.0	2.915090e+02	4.425750e+02	0.000000e+00	0.000000e+00	0.000000e+00	5.600000e+02	4.820000e+03
yr_built	21613.0	1.971005e+03	2.937341e+01	1.900000e+03	1.951000e+03	1.975000e+03	1.997000e+03	2.015000e+03
yr_renovated	21613.0	8.440226e+01	4.016792e+02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.015000e+03
zipcode	21613.0	9.807794e+04	5.350503e+01	9.800100e+04	9.803300e+04	9.806500e+04	9.811800e+04	9.819900e+04
lat	21613.0	4.756005e+01	1.385637e-01	4.715590e+01	4.747100e+01	4.757180e+01	4.767800e+01	4.777760e+01
long	21613.0	-1.222139e+02	1.408283e-01	-1.225190e+02	-1.223280e+02	-1.222300e+02	-1.221250e+02	-1.213150e+02
sqft_living15	21613.0	1.986552e+03	6.853913e+02	3.990000e+02	1.490000e+03	1.840000e+03	2.360000e+03	6.210000e+03
sqft_lot15	21613.0	1.276846e+04	2.730418e+04	6.510000e+02	5.100000e+03	7.620000e+03	1.008300e+04	8.712000e+05

```
sns.set(style="whitegrid", font_scale=1)

plt.figure(figsize=(13,13))
plt.title('Pearson Correlation Matrix',fontsize=25)
sns.heatmap(df.corr(),linewidths=0.25,vmax=0.7,square=True,cmap="GnBu",linecolor='w',
            annot=True, annot_kws={"size":7}, cbar_kws={"shrink": .7})
```

```
<ipython-input-21-1b9cab547edf>:5: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future
  sns.heatmap(df.corr(),linewidths=0.25,vmax=0.7,square=True,cmap="GnBu",linecolor='w',
<Axes: title={'center': 'Pearson Correlation Matrix'}>
```



	id	price	ooms	ooms	iving	ft_lot	loors	front	view	dition	rade	bove	ment	built	vated	code	lat	long	ng15	lot15
id	1	-0.017	0.0013	0.0052	-0.012	-0.13	0.019	-0.0027	0.012	-0.024	0.0081	-0.011	-0.0052	0.021	-0.017	-0.0082	-0.0019	0.021	-0.0029	-0.14
price	-0.017	1	0.31	0.53	0.7	0.09	0.26	0.27	0.4	0.036	0.67	0.61	0.32	0.054	0.13	-0.053	0.31	0.022	0.59	0.082
bedrooms	0.0013	0.31	1	0.52	0.58	0.032	0.18	-0.0066	0.08	0.028	0.36	0.48	0.3	0.15	0.019	-0.15	-0.0089	0.13	0.39	0.029
bathrooms	0.0052	0.53	0.52	1	0.75	0.088	0.5	0.064	0.19	-0.12	0.66	0.69	0.28	0.51	0.051	-0.2	0.025	0.22	0.57	0.087
sqft_living	-0.012	0.7	0.58	0.75	1	0.17	0.35	0.1	0.28	-0.059	0.76	0.88	0.44	0.32	0.055	-0.2	0.053	0.24	0.76	0.18
sqft_lot	-0.13	0.09	0.032	0.088	0.17	1	-0.0052	0.022	0.075	-0.009	0.11	0.18	0.015	0.053	0.0076	-0.13	-0.086	0.23	0.14	0.72
floors	0.019	0.26	0.18	0.5	0.35	-0.0052	1	0.024	0.029	-0.26	0.46	0.52	-0.25	0.49	0.0063	-0.059	0.05	0.13	0.28	-0.011
waterfront	-0.0027	0.27	-0.0066	0.064	0.1	0.022	0.024	1	0.4	0.017	0.083	0.072	0.081	-0.026	0.093	0.03	-0.014	-0.042	0.086	0.031
view	0.012	0.4	0.08	0.19	0.28	0.075	0.029	0.4	1	0.046	0.25	0.17	0.28	-0.053	0.1	0.085	0.0062	-0.078	0.28	0.073
condition	-0.024	0.036	0.028	-0.12	-0.059	-0.009	-0.26	0.017	0.046	1	-0.14	-0.16	0.17	-0.36	-0.061	0.003	-0.015	-0.11	-0.093	-0.0034
grade	0.0081	0.67	0.36	0.66	0.76	0.11	0.46	0.083	0.25	-0.14	1	0.76	0.17	0.45	0.014	-0.18	0.11	0.2	0.71	0.12
sqft_above	-0.011	0.61	0.48	0.69	0.88	0.18	0.52	0.072	0.17	-0.16	0.76	1	-0.052	0.42	0.023	-0.26	-0.00082	0.34	0.73	0.19
sqft_basement	-0.0052	0.32	0.3	0.28	0.44	0.015	-0.25	0.081	0.28	0.17	0.17	-0.052	1	-0.13	0.071	0.075	0.11	-0.14	0.2	0.017
yr_built	0.021	0.054	0.15	0.51	0.32	0.053	0.49	-0.026	-0.053	-0.36	0.45	0.42	-0.13	1	-0.22	-0.35	-0.15	0.41	0.33	0.071
yr_renovated	-0.017	0.13	0.019	0.051	0.055	0.0076	0.0063	0.093	0.1	-0.061	0.014	0.023	0.071	-0.22	1	0.064	0.029	-0.068	-0.0027	0.0079
zipcode	-0.0082	-0.053	-0.15	-0.2	-0.2	-0.13	-0.059	0.03	0.085	0.003	-0.18	-0.26	0.075	-0.35	0.064	1	0.27	-0.56	-0.28	-0.15
lat	-0.0019	0.31	-0.0089	0.025	0.053	-0.086	0.05	-0.014	0.0062	-0.015	0.11	-0.00082	0.11	-0.15	0.029	0.27	1	-0.14	0.049	-0.086
long	0.021	0.022	0.13	0.22	0.24	0.23	0.13	-0.042	-0.078	-0.11	0.2	0.34	-0.14	0.41	-0.068	-0.56	-0.14	1	0.33	0.25
sqft_living15	-0.0029	0.59	0.39	0.57	0.76	0.14	0.28	0.086	0.28	-0.093	0.71	0.73	0.2	0.33	-0.0027	-0.28	0.049	0.33	1	0.18
sqft_lot15	-0.14	0.082	0.029	0.087	0.18	0.72	-0.011	0.031	0.073	-0.0034	0.12	0.19	0.017	0.071	0.0079	-0.15	-0.086	0.25	0.18	1

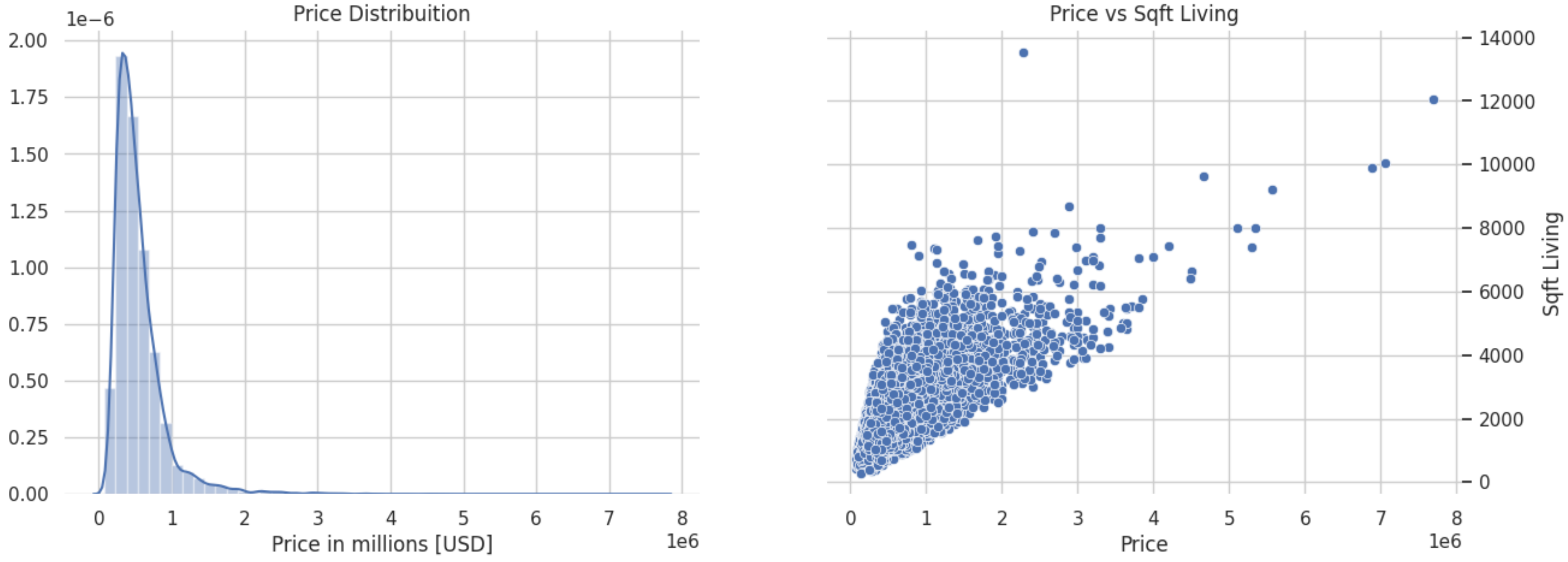


```
price_corr = df.corr()['price'].sort_values(ascending=False)
print(price_corr)
```

```
<ipython-input-22-7cbd0902dff7>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to
  price_corr = df.corr()['price'].sort_values(ascending=False)
price            1.000000
sqft_living      0.702035
grade            0.667434
sqft_above       0.605567
sqft_living15    0.585379
bathrooms        0.525138
view             0.397293
sqft_basement    0.323816
bedrooms         0.308350
lat              0.307003
waterfront       0.266369
floors           0.256794
yr_renovated     0.126434
sqft_lot         0.089661
sqft_lot15       0.082447
yr_built         0.054012
condition        0.036362
long             0.021626
id              -0.016762
zipcode         -0.053203
Name: price, dtype: float64
```

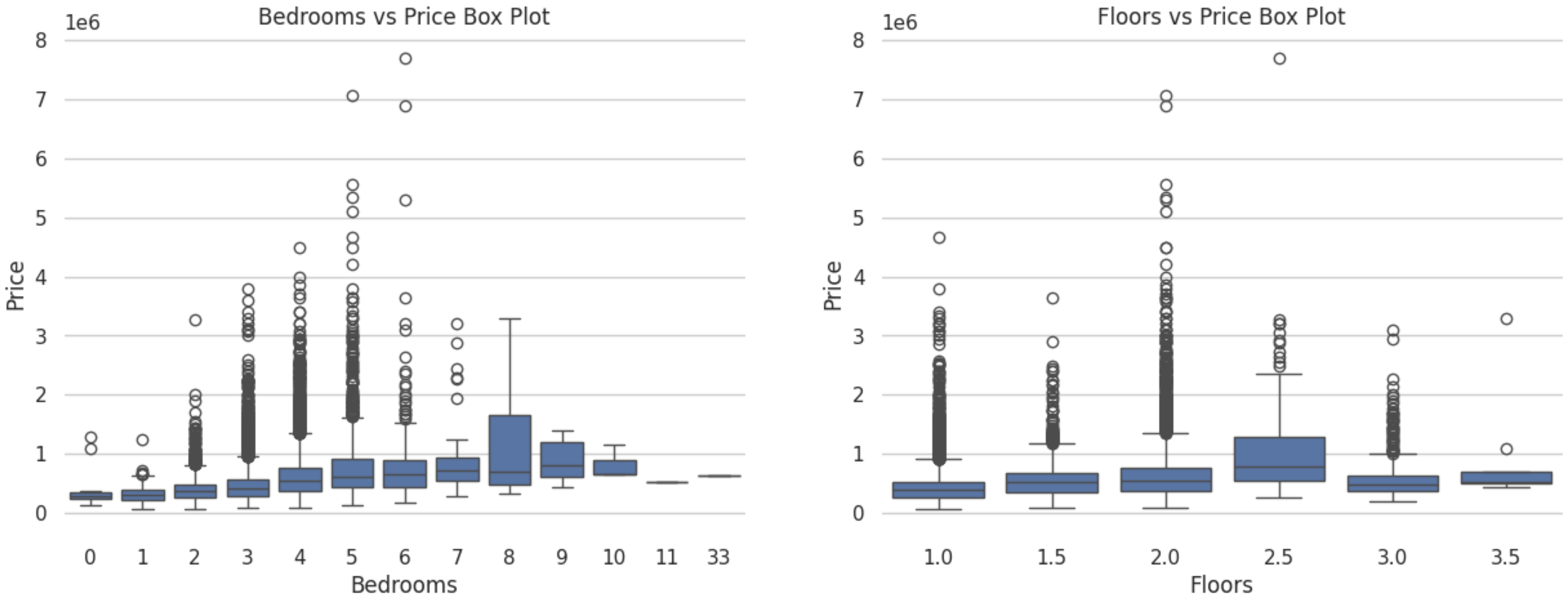
```
f, axes = plt.subplots(1, 2,figsize=(15,5))
sns.distplot(df['price'], ax=axes[0])
sns.scatterplot(x='price',y='sqft_living', data=df, ax=axes[1])
sns.despine(bottom=True, left=True)
axes[0].set(xlabel='Price in millions [USD]', ylabel='', title='Price Distribuition')
axes[1].set(xlabel='Price', ylabel='Sqft Living', title='Price vs Sqft Living')
axes[1].yaxis.set_label_position("right")
axes[1].yaxis.tick_right()
```

```
<ipython-input-23-8992c7a9a438>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['price'], ax=axes[0])
```



```
sns.set(style="whitegrid", font_scale=1)

f, axes = plt.subplots(1, 2,figsize=(15,5))
sns.boxplot(x=df['bedrooms'],y=df['price'], ax=axes[0])
sns.boxplot(x=df['floors'],y=df['price'], ax=axes[1])
sns.despine(bottom=True, left=True)
axes[0].set(xlabel='Bedrooms', ylabel='Price', title='Bedrooms vs Price Box Plot')
axes[1].set(xlabel='Floors', ylabel='Price', title='Floors vs Price Box Plot')
```

```
[Text(0.5, 0, 'Floors'),
 Text(0, 0.5, 'Price'),
 Text(0.5, 1.0, 'Floors vs Price Box Plot')]
```

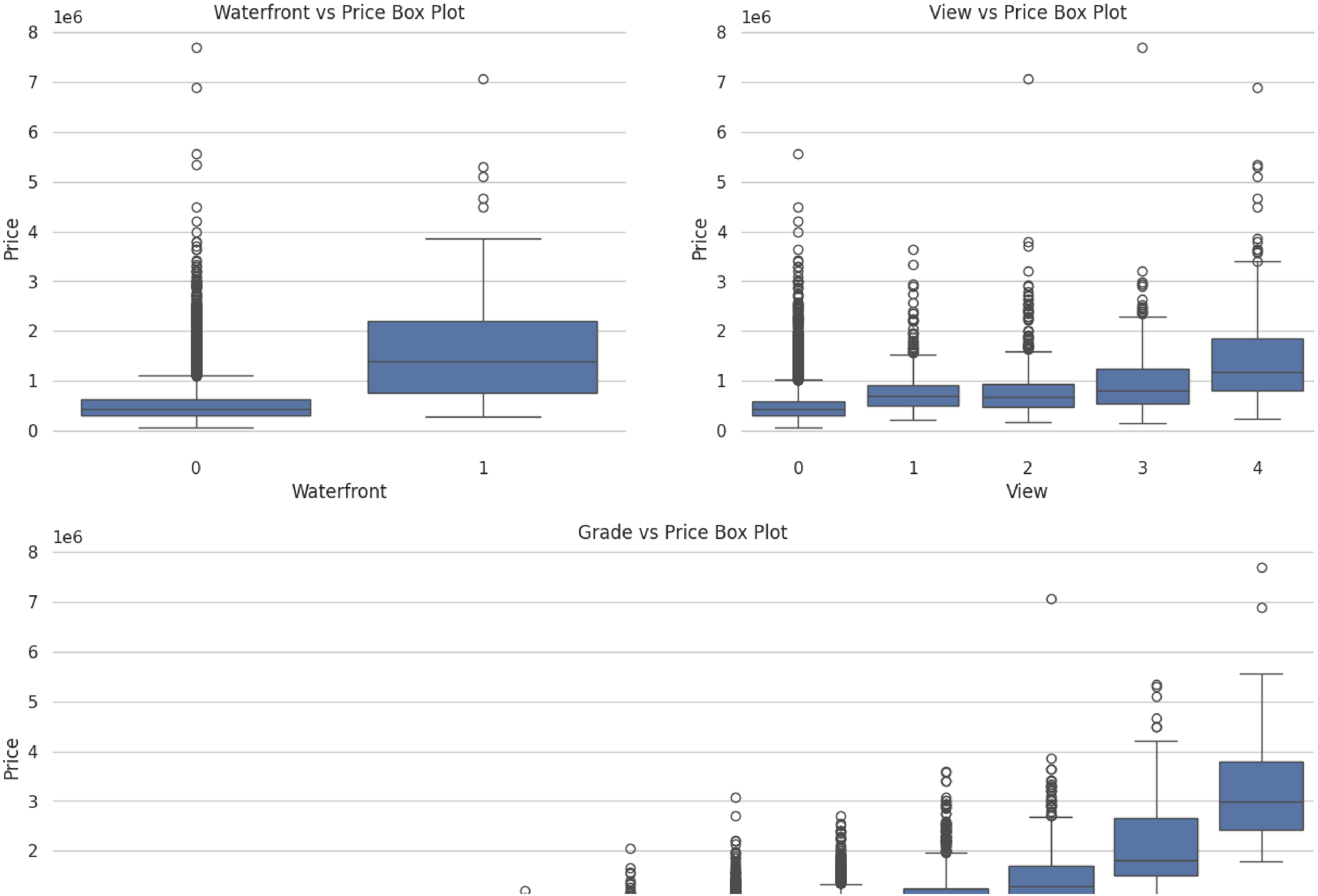


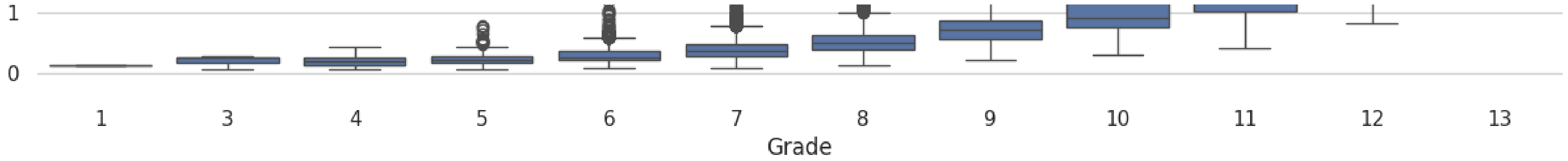
Bedrooms vs Price Box Plot

Floors vs Price Box Plot

```
f, axes = plt.subplots(1, 2,figsize=(15,5))
sns.boxplot(x=df['waterfront'],y=df['price'], ax=axes[0])
sns.boxplot(x=df['view'],y=df['price'], ax=axes[1])
sns.despine(left=True, bottom=True)
axes[0].set(xlabel='Waterfront', ylabel='Price', title='Waterfront vs Price Box Plot')
axes[1].set(xlabel='View', ylabel='Price', title='View vs Price Box Plot')

f, axe = plt.subplots(1, 1,figsize=(15,5))
sns.boxplot(x=df['grade'],y=df['price'], ax=axe)
sns.despine(left=True, bottom=True)
axe.set(xlabel='Grade', ylabel='Price', title='Grade vs Price Box Plot')
```

```
[Text(0.5, 0, 'Grade'),
 Text(0, 0.5, 'Price'),
 Text(0.5, 1.0, 'Grade vs Price Box Plot')]
```





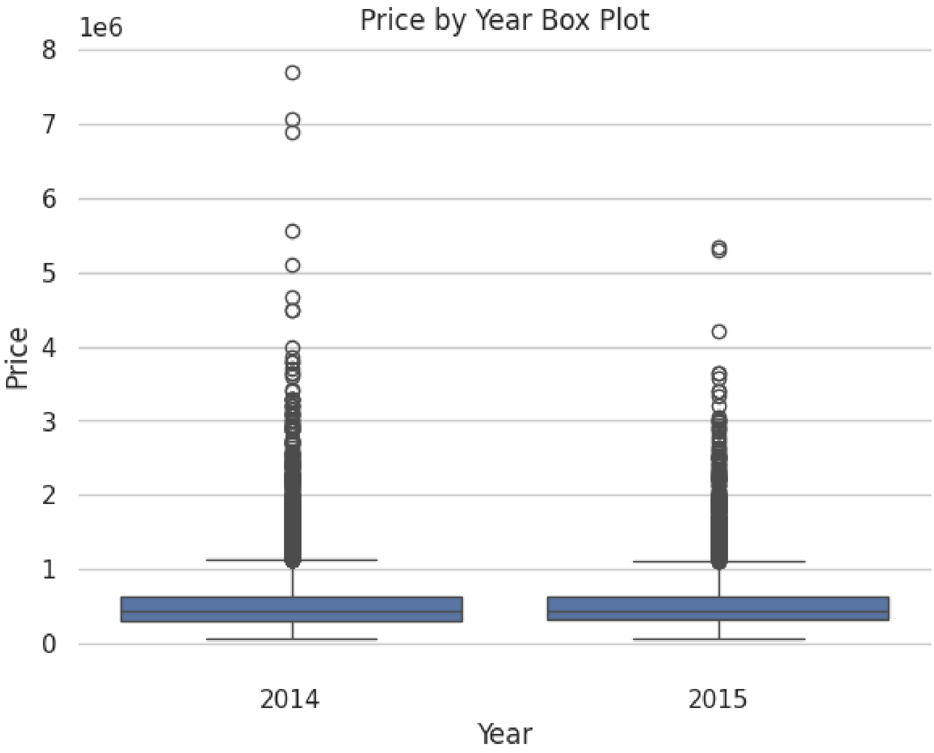
```
df = df.drop('id', axis=1)
df = df.drop('zipcode',axis=1)
```

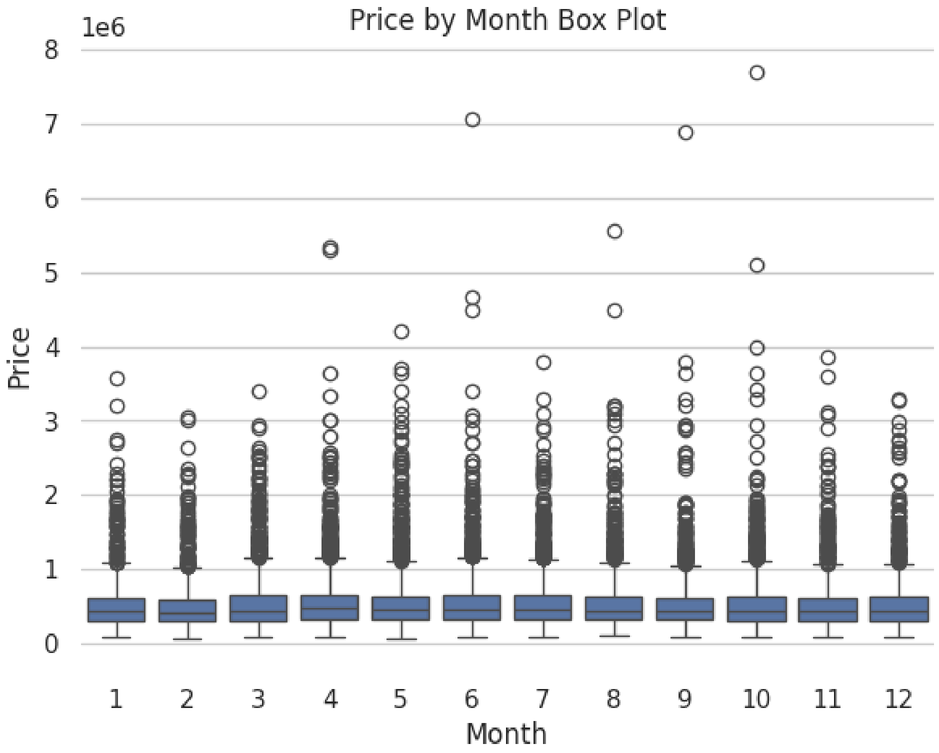
```
df['date'] = pd.to_datetime(df['date'])

df['month'] = df['date'].apply(lambda date:date.month)
df['year'] = df['date'].apply(lambda date:date.year)

df = df.drop('date',axis=1)

# Check the new columns
print(df.columns.values)
```

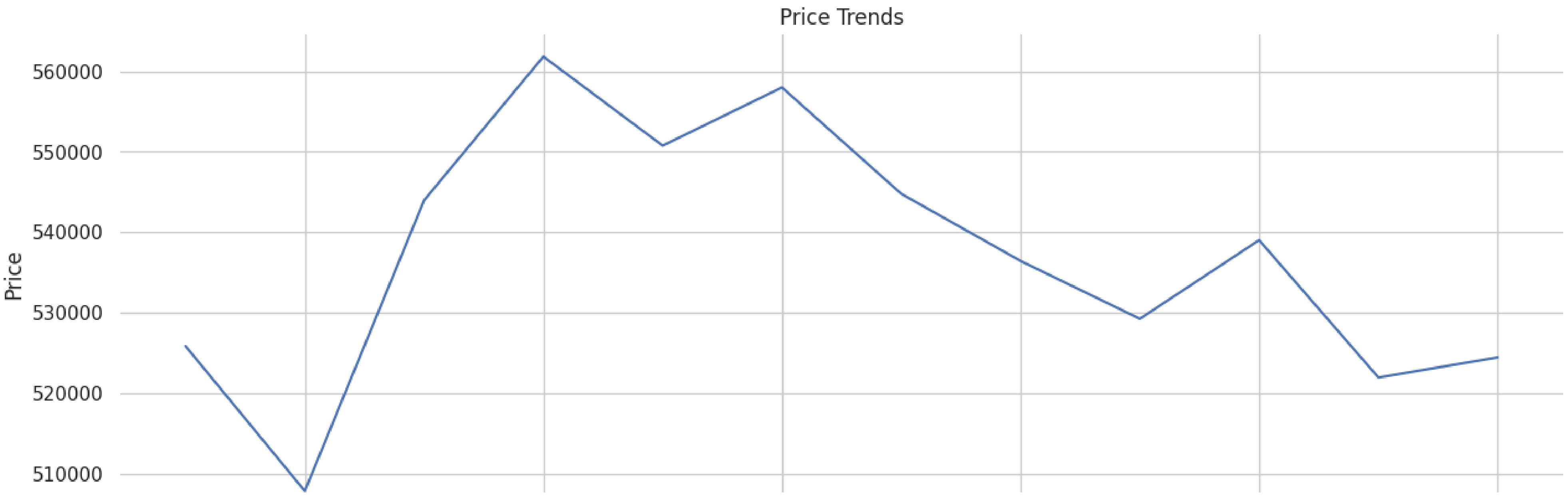
```
['price' 'bedrooms' 'bathrooms' 'sqft_living' 'sqft_lot' 'floors'
 'waterfront' 'view' 'condition' 'grade' 'sqft_above' 'sqft_basement'
 'yr_built' 'yr_renovated' 'lat' 'long' 'sqft_living15' 'sqft_lot15'
 'month' 'year']
```

```
f, axes = plt.subplots(1, 2,figsize=(15,5))
sns.boxplot(x='year',y='price',data=df, ax=axes[0])
sns.boxplot(x='month',y='price',data=df, ax=axes[1])
sns.despine(left=True, bottom=True)
axes[0].set(xlabel='Year', ylabel='Price', title='Price by Year Box Plot')
axes[1].set(xlabel='Month', ylabel='Price', title='Price by Month Box Plot')

f, axe = plt.subplots(1, 1,figsize=(15,5))
df.groupby('month').mean()['price'].plot()
sns.despine(left=True, bottom=True)
axe.set(xlabel='Month', ylabel='Price', title='Price Trends')
```

```
[Text(0.5, 0, 'Month'), Text(0, 0.5, 'Price'), Text(0.5, 1.0, 'Price Trends')]
```







```
# Features
X = df.drop('price',axis=1)

# Label
y = df['price']

# Split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=101)
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(15129, 19)
(6484, 19)
(15129,)
(6484,)
```

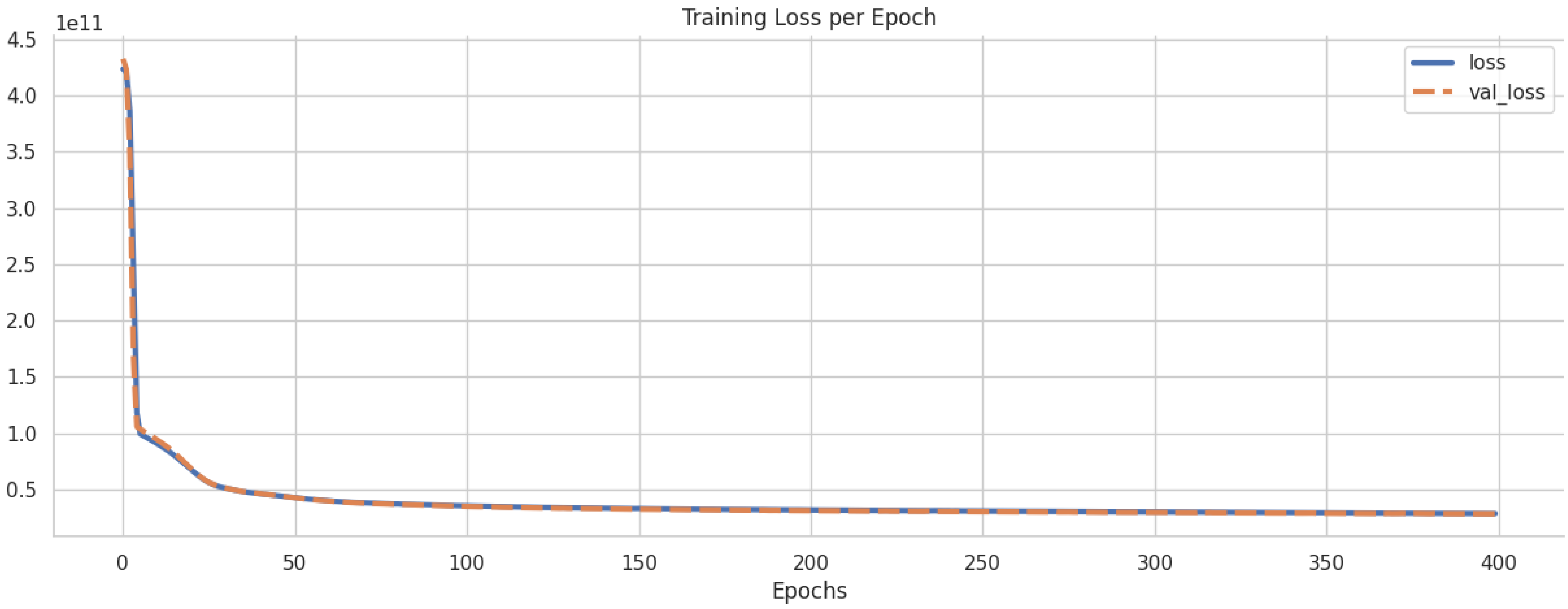
```
scaler = MinMaxScaler()

# fit and transfrom
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# everything has been scaled between 1 and 0
print('Max: ',X_train.max())
print('Min: ', X_train.min())
```

```
Max:  1.0000000000000002
Min:  0.0
```

```
model = Sequential()

# input layer
model.add(Dense(19,activation='relu'))

# hidden layers
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))

# output layer
model.add(Dense(1))

model.compile(optimizer='adam',loss='mse')


model.fit(x=X_train,y=y_train.values,
          validation_data=(X_test,y_test.values),
          batch_size=128,epochs=400)
```

```
119/119 [==============================] - 0s 4ms/step - loss: 28519084032.0000 - val_loss: 28249851904.0000
Epoch 387/400
119/119 [==============================] - 0s 3ms/step - loss: 28461717504.0000 - val_loss: 28227166208.0000
Epoch 388/400
119/119 [==============================] - 0s 3ms/step - loss: 28447885312.0000 - val_loss: 28236111872.0000
Epoch 389/400
119/119 [==============================] - 0s 4ms/step - loss: 28458059776.0000 - val_loss: 28177870848.0000
Epoch 390/400
119/119 [==============================] - 0s 4ms/step - loss: 28454924288.0000 - val_loss: 28218097664.0000
Epoch 391/400
119/119 [==============================] - 0s 3ms/step - loss: 28471775232.0000 - val_loss: 28149301248.0000
Epoch 392/400
119/119 [==============================] - 0s 3ms/step - loss: 28431503360.0000 - val_loss: 28182046720.0000
Epoch 393/400
119/119 [==============================] - 0s 4ms/step - loss: 28421937152.0000 - val_loss: 28150104064.0000
Epoch 394/400
119/119 [==============================] - 0s 4ms/step - loss: 28381530112.0000 - val_loss: 28120223744.0000
Epoch 395/400
119/119 [==============================] - 0s 4ms/step - loss: 28387883008.0000 - val_loss: 28121425920.0000
Epoch 396/400
119/119 [==============================] - 0s 4ms/step - loss: 28371150848.0000 - val_loss: 28178825216.0000
Epoch 397/400
119/119 [==============================] - 0s 3ms/step - loss: 28336060416.0000 - val_loss: 28081702912.0000
Epoch 398/400
119/119 [==============================] - 0s 4ms/step - loss: 28331995136.0000 - val_loss: 28110159872.0000
Epoch 399/400
119/119 [==============================] - 1s 5ms/step - loss: 28299653120.0000 - val_loss: 28048721920.0000
Epoch 400/400
119/119 [==============================] - 1s 5ms/step - loss: 28302182400.0000 - val_loss: 28045254656.0000
<keras.src.callbacks.History at 0x7dd02f5f89d0>
```

```
losses = pd.DataFrame(model.history.history)

plt.figure(figsize=(15,5))
sns.lineplot(data=losses,lw=3)
plt.xlabel('Epochs')
plt.ylabel('')
plt.title('Training Loss per Epoch')
sns.despine()
```



```
# predictions on the test set
predictions = model.predict(X_test)

print('MAE: ',mean_absolute_error(y_test,predictions))
print('MSE: ',mean_squared_error(y_test,predictions))
print('RMSE: ',np.sqrt(mean_squared_error(y_test,predictions)))
print('Variance Regression Score: ',explained_variance_score(y_test,predictions))

print('\n\nDescriptive Statistics:\n',df['price'].describe())
```

```
203/203 [==============================] - 2s 7ms/step
MAE:  104748.00371045554
MSE:  28045253475.152184
RMSE:  167467.17133561487
Variance Regression Score:  0.8000695603775322


Descriptive Statistics:
```

```
         count    2.161300e+04
        mean     5.400881e+05
        std      3.671272e+05
        min      7.500000e+04
        25%      3.219500e+05
        50%      4.500000e+05
        75%      6.450000e+05
        max      7.700000e+06
        Name: price, dtype: float64
```

```
f, axes = plt.subplots(1, 2,figsize=(15,5))

# Our model predictions
plt.scatter(y_test,predictions)

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

errors = y_test.values.reshape(6484, 1) - predictions
sns.distplot(errors, ax=axes[0])

sns.despine(left=True, bottom=True)
axes[0].set(xlabel='Error', ylabel='', title='Error Histogram')
axes[1].set(xlabel='Test True Y', ylabel='Model Predictions', title='Model Predictions vs Perfect Fit')
```

```
     <ipython-input-36-83b90cb0bedd>:10: UserWarning:

     `distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```