githosted / **Phase2Pro** Public

forked from learn-co-curriculum/dsc-phase-2-project

Code Issues Pull requests Actions Projects Wiki Security Insig

main

Phase2Pro / REDO1.ipynb

githosted push cleaner redo with preds, log trans, edited - History

1 contributor

3236 lines (3236 sloc) | 556 KB

In [1]:
```
#Import packages
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
import scipy.stats as stats
from scipy import stats
from statsmodels.formula.api import ols
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from mpl_toolkits import mplot3d
import sklearn.metrics as metrics
import random
from math import sqrt
import seaborn as sns
plt.style.use('seaborn')
```

In [2]:
```
#Import data set
data = pd.read_csv('data\\kc_house_data.csv')
```

In [3]:
```
#inspect shape
data.shape
```

Out[3]:
```
(21597, 21)
```

In [4]:
```
#Review dataset
data.head()
```

Out[4]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wate

0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0

5 rows × 21 columns

In [5]:
```
#review null values
data.isnull().sum()
```

Out[5]:
```
id                0
date              0
price             0
bedrooms          0
bathrooms         0
sqft_living       0
sqft_lot          0
floors            0
waterfront     2376
view             63
condition         0
grade             0
sqft_above        0
sqft_basement     0
yr_built          0
yr_renovated   3842
zipcode           0
lat               0
long              0
sqft_living15     0
sqft_lot15        0
dtype: int64
```

After reviewing the business problem and the dataset categories, I have removed unnecessary columns

In [6]:
```
#drop unnecessary column data
data.drop(['id', 'date', 'waterfront', 'sqft_above', 'sqft_basement', 'lat', 'l
```

In [7]:
```
#create the column data
continuous = ['price', 'sqft_lot', 'sqft_living']
categoricals = ['bedrooms', 'bathrooms', 'floors', 'condition','yr_built', 'zip
data_cont = data[continuous]
```

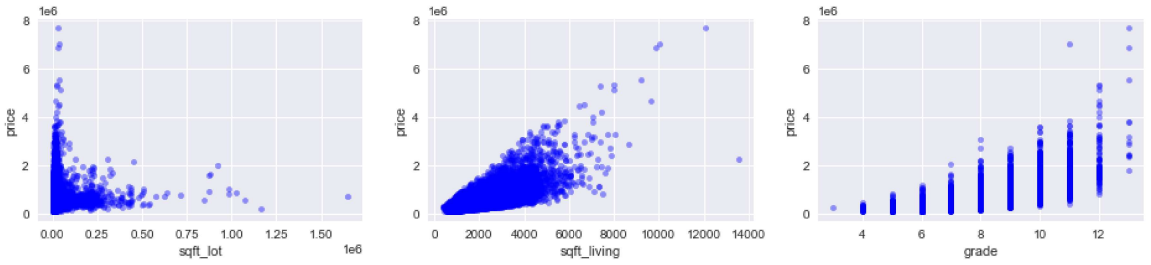
In [8]:
```
#checking to see if there are unique values
data.nunique()
```

Out[8]:
```
price          3622
bedrooms         12
```

```
bathrooms        29
sqft_living    1034
sqft_lot       9776
floors            6
condition         5
grade            11
yr_built        116
zipcode          70
dtype: int64
```

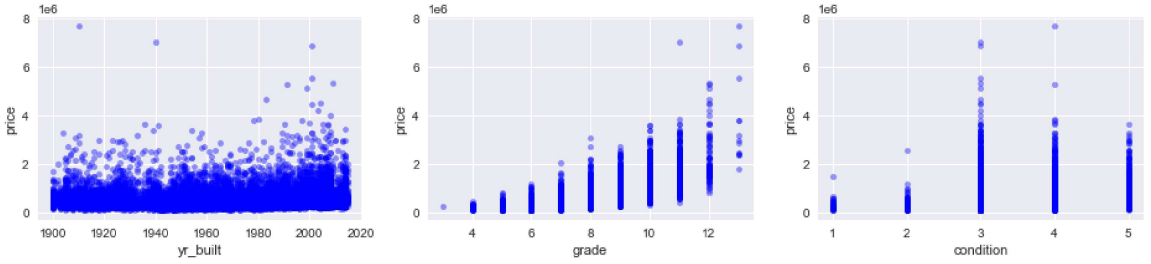
In [9]:

```
#Continuous data visualisations
import matplotlib.pyplot as plt
%matplotlib inline

fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(16,3))

for xcol, ax in zip(['sqft_lot', 'sqft_living', 'grade'], axes):
    data.plot(kind='scatter', x=xcol, y='price', ax=ax, alpha=0.4, color='b')
```



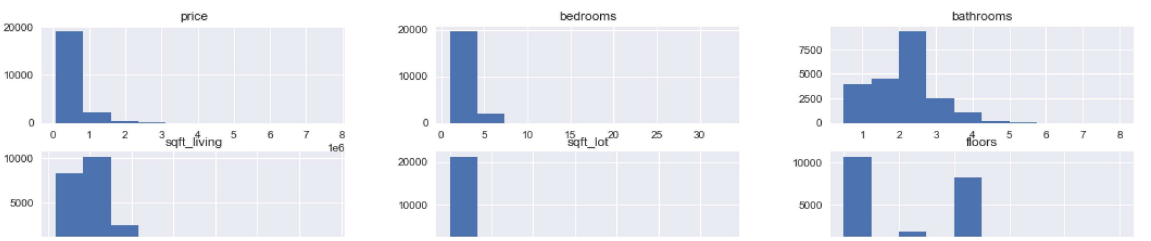
In [10]:

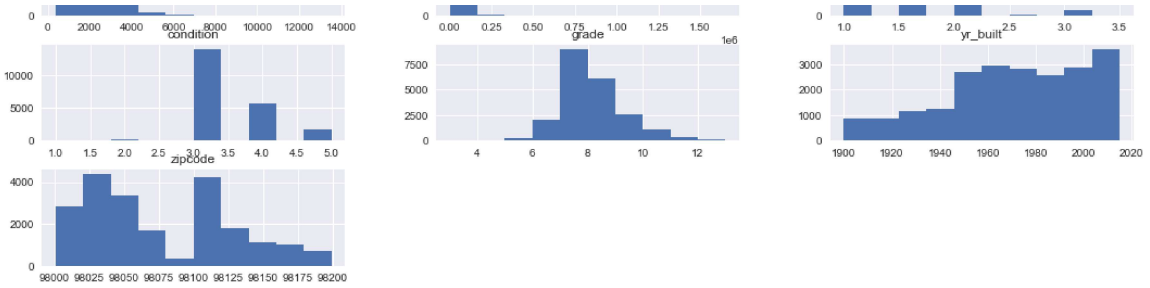
```
# data visualisations for linearity
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(16,3))

for xcol, ax in zip([ 'yr_built', 'grade', 'condition'], axes):
    data.plot(kind='scatter', x=xcol, y='price', ax=ax, alpha=0.4, color='b')
```



In [11]:

```
#Useful histogram to view any variances and understand the data set
import warnings
warnings.filterwarnings('ignore')
fig = plt.figure(figsize = (18,8))
ax = fig.gca()
data.hist(ax = ax);
```





In [12]:
```
# log features declared with lamda values to convert int to floats
log_names = [f'{column}_log' for column in data_cont.columns]
pd.options.display.float_format = lambda x : '{:.0f}'.format(x) if int(x) == x
data_log = np.log(data_cont)
data_log.columns = log_names
```

In [13]:
```
# normalize continued features
def normalize(feature):
    return (feature - feature.mean()) / feature.std()
data_log_norm = data_log.apply(normalize)
```

In [14]:
```
#creating duimmy data
pd.get_dummies(data)
```

Out[14]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	yr_built
0	221900	3	1	1180	5650	1	3	7	1955
1	538000	3	2.25	2570	7242	2	3	7	1951
2	180000	2	1	770	10000	1	3	6	1933
3	604000	4	3	1960	5000	1	5	7	1965
4	510000	3	2	1680	8080	1	3	8	1987
...	...	...	...	...	...	...	...	...	...
21592	360000	3	2.50	1530	1131	3	3	8	2009
21593	400000	4	2.50	2310	5813	2	3	8	2014
21594	402101	2	0.75	1020	1350	2	3	7	2009
21595	400000	3	2.50	1600	2388	2	3	8	2004
21596	325000	2	0.75	1020	1076	2	3	7	2008

21597 rows × 10 columns

In [15]:
```
#declaring categorical dummy data
data_ohe = pd.get_dummies(data[categoricals], drop_first=True)
```

In [16]:
```
#combining two variables
preprocessed = pd.concat([data_log_norm, data_ohe], axis=1)
```

```
preprocessed.head()
```

Out[16]:

	price_log	sqft_lot_log	sqft_living_log	bedrooms	bathrooms	floors	condition	yr_built	z
0	-1.40	-0.39	-1.13	3	1	1	3	1955	
1	0.28	-0.11	0.71	3	2.25	2	3	1951	
2	-1.80	0.24	-2.13	2	1	1	3	1933	
3	0.50	-0.52	0.07	4	3	1	5	1965	
4	0.18	0.01	-0.29	3	2	1	3	1987	

In [17]:

```
#$$ \hat{price} = \hat{\beta}_{0} + \hat{\beta}_{1}lotsize + \hat{\beta}_{2}
```

In [18]:

```
# one hot encode (ohe)categoricals
data_ohe = pd.get_dummies(data[categoricals], prefix=categoricals[0], drop_firs
preprocessed = pd.concat([data_log_norm, data_ohe], axis=1)
X = preprocessed.drop('grade', axis=1)
y = preprocessed['grade']
```

In [19]:

```
# Split the data into training and test sets (assign 20% to test set)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

In [20]:

```
# A brief preview of train-test split to create test training data and dummy da
print(len(X_train), len(X_test), len(y_train), len(y_test))
```

```
17277 4320 17277 4320
```

In [21]:

```
#apply model to the train set
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, y_train)
y_hat_test = linreg.predict(X_test)
```

In [22]:

```
#calculate training and test MSE
from sklearn.metrics import mean_squared_error
test_residuals = y_hat_test - y_test
test_mse = mean_squared_error(y_test, y_hat_test)
test_mse
```

Out[22]:

```
0.4051496416650118
```

In [23]:

```
from sklearn.metrics import mean_squared_error, make_scorer
from sklearn.model_selection import cross_val_score
mse = make_scorer(mean_squared_error)
cv_5_results = cross_val_score(linreg, X, y, cv=5, scoring=mse)
```

In [24]:

```
cv_5_results.mean()
```

Out[24]:
```
0.3986158452807703
```

In [25]:
```
mlr = smf.ols(formula="price ~ sqft_living + sqft_lot", data=data).fit()
print(mlr.params)
```

```
Intercept     -44,331.93
sqft_living       283.14
sqft_lot           -0.29
dtype: float64
```

In [26]:
```
#stipulating x & y variables
X = preprocessed.drop('price_log', axis=1)
y = preprocessed['price_log']
```

In [27]:
```
#OLS squared data representative of three variables to predict
outcome = 'price'
x_cols = ['grade', 'sqft_living', 'sqft_lot']
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model = ols(formula=formula, data=data).fit()
model.summary()
```

Out[27]:

OLS Regression Results

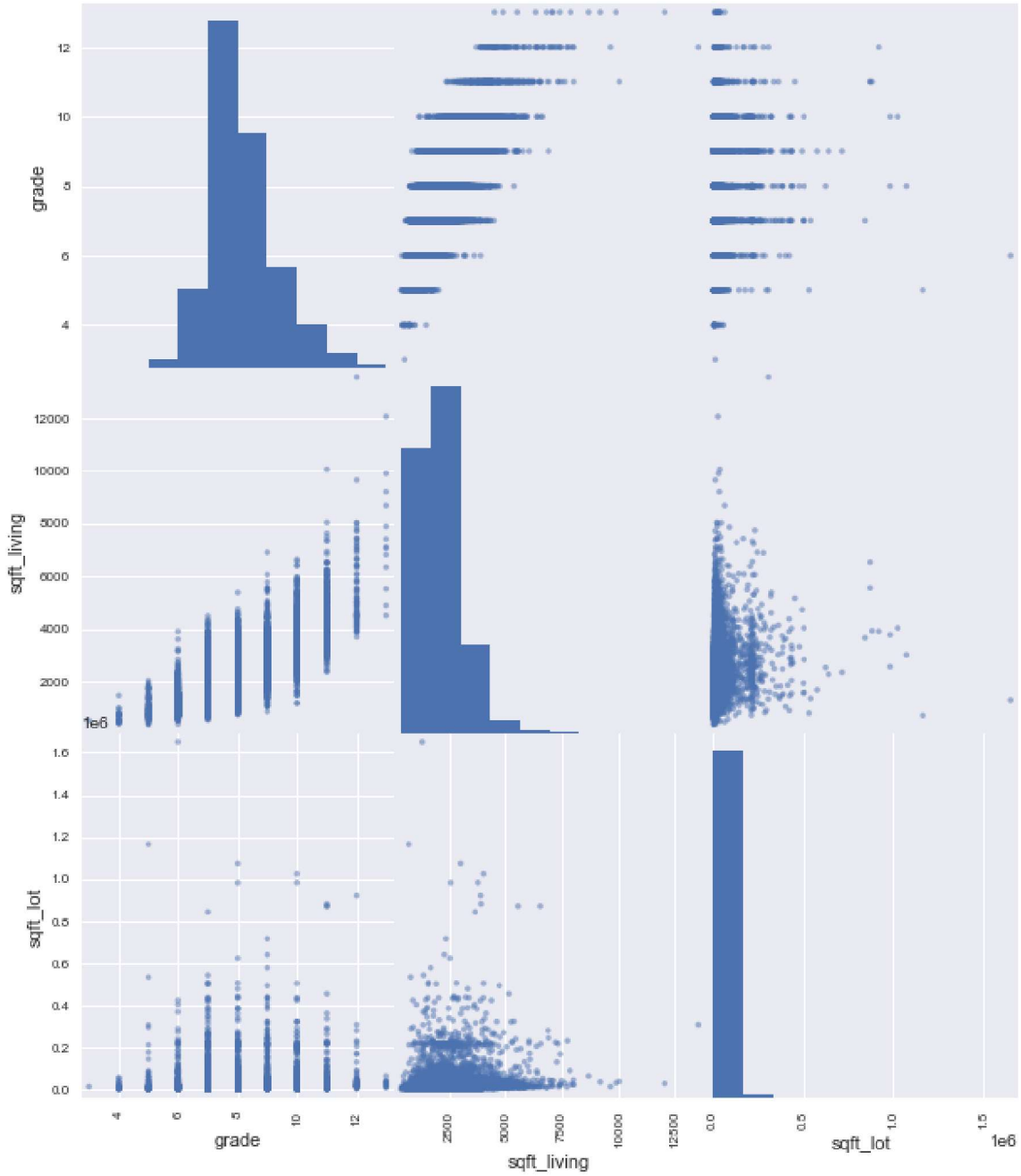
Dep. Variable:	price	**R-squared:**	0.535
Model:	OLS	**Adj. R-squared:**	0.535
Method:	Least Squares	**F-statistic:**	8295.
Date:	Fri, 03 Jun 2022	**Prob (F-statistic):**	0.00
Time:	15:34:05	**Log-Likelihood:**	-2.9911e+05
No. Observations:	21597	**AIC:**	5.982e+05
Df Residuals:	21593	**BIC:**	5.983e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-6.011e+05	1.33e+04	-45.074	0.000	-6.27e+05	-5.75e+05
grade	9.889e+04	2246.967	44.012	0.000	9.45e+04	1.03e+05
sqft_living	186.3532	2.896	64.344	0.000	180.676	192.030
sqft_lot	-0.2407	0.042	-5.760	0.000	-0.323	-0.159

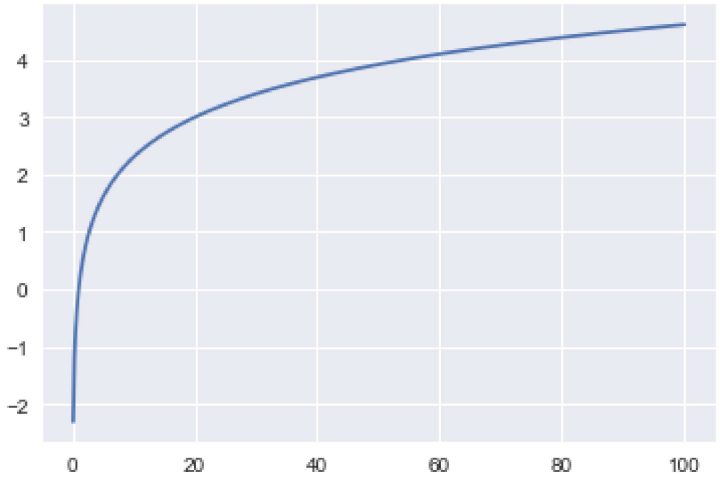
Omnibus:	16895.228	**Durbin-Watson:**	1.975
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	991094.879
Skew:	3.287	**Prob(JB):**	0.00
Kurtosis:	35.529	**Cond. No.**	3.49e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.49e+05. This might indicate that there are strong multicollinearity or other numerical problems.

In [28]:
```
#visualising matrices
pd.plotting.scatter_matrix(data[x_cols], figsize=(10,12));
```



In [29]:
```
#Log trandsformation visual representation
x = np.linspace(start=-100, stop=100, num=10**3)
y = np.log(x)
plt.plot(x, y);
```



In [30]:

```
#Predicting data values
data_preds = data.drop('price', axis=1)
data_target = data['price']
data_preds.head()
```

Out[30]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	yr_built	zipcode
0	3	1	1180	5650	1	3	7	1955	98178
1	3	2.25	2570	7242	2	3	7	1951	98125
2	2	1	770	10000	1	3	6	1933	98028
3	4	3	1960	5000	1	5	7	1965	98136
4	3	2	1680	8080	1	3	8	1987	98074

In [31]:

```
#use sm,add_constant() to add constant term/y-intercept
#optimise betas adding another predictor intercept term = bo x const
predictors = sm.add_constant(data_preds)
predictors
```

Out[31]:

	const	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	yr_built
0	1	3	1	1180	5650	1	3	7	1955
1	1	3	2.25	2570	7242	2	3	7	1951
2	1	2	1	770	10000	1	3	6	1933
3	1	4	3	1960	5000	1	5	7	1965
4	1	3	2	1680	8080	1	3	8	1987
...	...	...	...	...	...	...	...	...	...
21592	1	3	2.50	1530	1131	3	3	8	2009
21593	1	4	2.50	2310	5813	2	3	8	2014
21594	1	2	0.75	1020	1350	2	3	7	2009
21595	1	3	2.50	1600	2388	2	3	8	2004

21596	1	2	0.75	1020	1076	2	3	7	2008

21597 rows × 10 columns

In [32]:
```
model = sm.OLS(data_target, predictors).fit()
model.summary()
```

Out[32]: OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.618
Model:	OLS	**Adj. R-squared:**	0.618
Method:	Least Squares	**F-statistic:**	3880.
Date:	Fri, 03 Jun 2022	**Prob (F-statistic):**	0.00
Time:	15:34:06	**Log-Likelihood:**	-2.9700e+05
No. Observations:	21597	**AIC:**	5.940e+05
Df Residuals:	21587	**BIC:**	5.941e+05
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	7.328e+04	3.17e+06	0.023	0.982	-6.15e+06	6.3e+06
bedrooms	-4.886e+04	2127.161	-22.967	0.000	-5.3e+04	-4.47e+04
bathrooms	5.263e+04	3588.968	14.664	0.000	4.56e+04	5.97e+04
sqft_living	187.8342	3.426	54.824	0.000	181.119	194.550
sqft_lot	-0.2377	0.038	-6.194	0.000	-0.313	-0.162
floors	2.028e+04	3621.615	5.599	0.000	1.32e+04	2.74e+04
condition	2.028e+04	2601.136	7.795	0.000	1.52e+04	2.54e+04
grade	1.311e+05	2238.651	58.560	0.000	1.27e+05	1.35e+05
yr_built	-3954.9216	73.722	-53.647	0.000	-4099.422	-3810.422
zipcode	69.5884	31.814	2.187	0.029	7.231	131.946

Omnibus:	17321.831	**Durbin-Watson:**	1.984
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1210280.235
Skew:	3.359	**Prob(JB):**	0.00
Kurtosis:	39.053	**Cond. No.**	2.04e+08

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.04e+08. This might indicate that there are strong multicollinearity or other numerical problems.

In [33]:
```
data_preds_scaled = (data_preds - np.mean(data_preds)) /np.std(data_preds)
pd.options.display.float_format = lambda x : '{:.0f}'.format(x) if int(x) == x
```

In [34]:
```
data_preds_scaled.describe()
```

Out[34]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	yr_built	zipcode
count	21597	21597	21597	21597	21597	21597	21597	21597	21597
mean	0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	0.00	-0.00
std	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
min	-2.56	-2.10	-1.86	-0.35	-0.92	-3.70	-3.97	-2.42	-1.44
25%	-0.40	-0.48	-0.71	-0.24	-0.92	-0.63	-0.56	-0.68	-0.84
50%	-0.40	0.17	-0.19	-0.18	0.01	-0.63	-0.56	0.14	-0.24
75%	0.68	0.50	0.51	-0.11	0.94	0.91	0.29	0.89	0.75
max	31.98	7.65	12.48	39.51	3.72	2.44	4.55	1.50	2.26

In [35]:
```
predictors = sm.add_constant(data_preds_scaled)
model = sm.OLS(data_target, predictors).fit()
model.summary()
```

Out[35]: OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.618
Model:	OLS	**Adj. R-squared:**	0.618
Method:	Least Squares	**F-statistic:**	3880.
Date:	Fri, 03 Jun 2022	**Prob (F-statistic):**	0.00
Time:	15:34:06	**Log-Likelihood:**	-2.9700e+05
No. Observations:	21597	**AIC:**	5.940e+05
Df Residuals:	21587	**BIC:**	5.941e+05
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.403e+05	1545.416	349.612	0.000	5.37e+05	5.43e+05
bedrooms	-4.525e+04	1970.341	-22.967	0.000	-4.91e+04	-4.14e+04
bathrooms	4.047e+04	2759.796	14.664	0.000	3.51e+04	4.59e+04
sqft_living	1.724e+05	3145.471	54.824	0.000	1.66e+05	1.79e+05

sqft_lot	-9844.1900	1589.325	-6.194	0.000	-1.3e+04	-6728.996
floors	1.094e+04	1954.478	5.599	0.000	7111.314	1.48e+04
condition	1.319e+04	1692.118	7.795	0.000	9874.150	1.65e+04
grade	1.538e+05	2626.324	58.560	0.000	1.49e+05	1.59e+05
yr_built	-1.162e+05	2165.543	-53.647	0.000	-1.2e+05	-1.12e+05
zipcode	3723.8038	1702.419	2.187	0.029	386.936	7060.672

Omnibus:	17321.831	**Durbin-Watson:**	1.984
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1210280.235
Skew:	3.359	**Prob(JB):**	0.00
Kurtosis:	39.053	**Cond. No.**	4.83

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [36]:
```
#Transforming dataset
ss = StandardScaler()
ss.fit(data_preds)
data_preds_st_scaled = ss.transform(data_preds_scaled)
```

In [37]:
```
#boolean value declared
np.allclose(data_preds_st_scaled, data_preds_scaled)
```

Out[37]:
```
False
```

In [38]:
```
#preview the predicted df
data_preds_scaled.head()
```

Out[38]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	yr_built	zipcode
0	-0.40	-1.45	-0.98	-0.23	-0.92	-0.63	-0.56	-0.54	1.87
1	-0.40	0.17	0.53	-0.19	0.94	-0.63	-0.56	-0.68	0.88
2	-1.48	-1.45	-1.43	-0.12	-0.92	-0.63	-1.41	-1.29	-0.93
3	0.68	1.15	-0.13	-0.24	-0.92	2.44	-0.56	-0.20	1.08
4	-0.40	-0.15	-0.44	-0.17	-0.92	-0.63	0.29	0.54	-0.07

In [39]:
```
#obtaining statistic values of target mean
data_target.mean()
```

Out[39]:
```
540296.5735055795
```

In [40]:
```
#created array
data_preds_st_scaled[:5, :]
```

Out[40]:
```
array([[-4.07664356e+00, -4.63856226e+00, -2.26700447e+00,
        -3.64622677e-01, -4.46503867e+00, -6.21002614e+00,
        -7.00554752e+00, -6.71174225e+01, -1.83279241e+03],
       [-4.07664356e+00, -2.52460998e+00, -2.26535536e+00,
        -3.64621749e-01, -1.03149188e+00, -6.21002614e+00,
        -7.00554752e+00, -6.71220582e+01, -1.83281092e+03],
       [-5.24215843e+00, -4.63856226e+00, -2.26749090e+00,
        -3.64620141e-01, -4.46503867e+00, -6.21002614e+00,
        -7.73211549e+00, -6.71429189e+01, -1.83284479e+03],
       [-2.91112870e+00, -1.25623861e+00, -2.26607907e+00,
        -3.64623056e-01, -4.46503867e+00, -1.48401687e+00,
        -7.00554752e+00, -6.71058332e+01, -1.83280708e+03],
       [-4.07664356e+00, -2.94740043e+00, -2.26641127e+00,
        -3.64621260e-01, -4.46503867e+00, -6.21002614e+00,
        -6.27897955e+00, -6.70803367e+01, -1.83282873e+03]])
```

In [41]:
```
#confirm lr to run
mlr = LinearRegression()
mlr.fit(data_preds_st_scaled, data_target)
```

Out[41]:
```
LinearRegression()
```

In [42]:
```
#setting co-efficient array
mlr.coef_
```

Out[42]:
```
array([-4.19174557e+04,  3.11190790e+04,  1.58321622e+08, -4.07664426e+08,
        5.90519890e+03,  8.58103632e+03,  1.80431342e+05, -3.41256111e+06,
        1.99267568e+05])
```

In [43]:
```
#calculating intercept data value
mlr.intercept_
```

Out[43]:
```
348061654.01133776
```

In [44]:
```
#calculating linear regression score
mlr.score(data_preds_st_scaled, data_target)
```

Out[44]:
```
0.6179675998938288
```

In [45]:
```
#validating though y^test
y_hat = mlr.predict(data_preds_st_scaled)
y_hat
```

Out[45]:
```
array([298603.03598171, 657505.59607589, 214886.76505136, ...,
       109614.32152653, 404282.31094038, 113634.37707627])
```

In [46]:
```
#find values for array
data_preds_st_scaled.shape
```

Out[46]:
```
(21597, 9)
```

In [47]:
```
#staged array
base_pred = np.zeros(9).reshape(1, -1)
base_pred
```

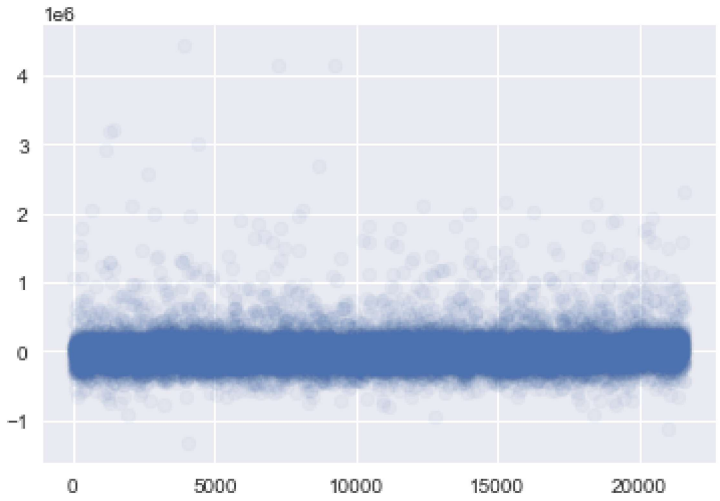
Out[47]:
```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

In [48]:
```
#predicted array
mlr.predict(base_pred)
```

Out[48]:
```
array([3.48061654e+08])
```

In [49]:
```
#Review y hat residual
y_hat = mlr.predict(data_preds_st_scaled)
resid = (data_target - y_hat)

plt.scatter(x=range(y_hat.shape[0]),y=resid, alpha=0.05)
```

Out[49]:
```
<matplotlib.collections.PathCollection at 0x15d9de2f370>
```



In [50]:
```
#overall metrics predicted data target metric to be .619 or 62%
metrics.r2_score(data_target, mlr.predict(data_preds_st_scaled))
```

Out[50]:
```
0.6179675998938288
```

In [51]:
```
#setting predictors location
data_pred = data.iloc[:,0:12]
data_pred.head()
```

Out[51]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	yr_built	zip
0	221900	3	1	1180	5650	1	3	7	1955	9
1	538000	3	2.25	2570	7242	2	3	7	1951	9
2	180000	2	1	770	10000	1	3	6	1933	9
3	604000	4	3	1960	5000	1	5	7	1965	9

4	510000	3	2	1680	8080	1	3	8	1987	9

In [52]:
```
#visualse value of correlation percentage
data_pred.corr()
```

Out[52]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	yr_bu
price	1	0.31	0.53	0.70	0.09	0.26	0.04	0.67	0.
bedrooms	0.31	1	0.51	0.58	0.03	0.18	0.03	0.36	0.
bathrooms	0.53	0.51	1	0.76	0.09	0.50	-0.13	0.67	0.
sqft_living	0.70	0.58	0.76	1	0.17	0.35	-0.06	0.76	0.
sqft_lot	0.09	0.03	0.09	0.17	1	-0.00	-0.01	0.11	0.
floors	0.26	0.18	0.50	0.35	-0.00	1	-0.26	0.46	0.
condition	0.04	0.03	-0.13	-0.06	-0.01	-0.26	1	-0.15	-0.
grade	0.67	0.36	0.67	0.76	0.11	0.46	-0.15	1	0.
yr_built	0.05	0.16	0.51	0.32	0.05	0.49	-0.36	0.45	
zipcode	-0.05	-0.15	-0.20	-0.20	-0.13	-0.06	0.00	-0.19	-0.

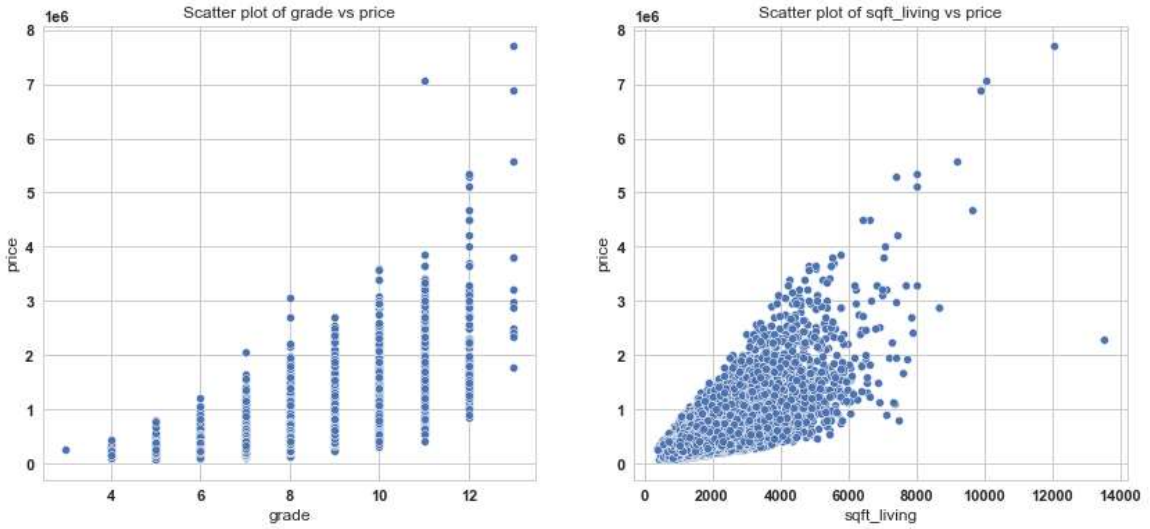
In [53]:
```
#check for connections
abs(data_pred.corr()) >= 0.70
```

Out[53]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	yr_bu
price	True	False	False	True	False	False	False	False	Fa
bedrooms	False	True	False	False	False	False	False	False	Fa
bathrooms	False	False	True	True	False	False	False	False	Fa
sqft_living	True	False	True	True	False	False	False	True	Fa
sqft_lot	False	False	False	False	True	False	False	False	Fa
floors	False	False	False	False	False	True	False	False	Fa
condition	False	False	False	False	False	False	True	False	Fa
grade	False	False	False	True	False	False	False	True	Fa
yr_built	False	False	False	False	False	False	False	False	Tr
zipcode	False	False	False	False	False	False	False	False	Fa

In [54]:
```
#Compare linearity between 2 variables for further analysis without exaggeratin
sns.set_theme(color_codes=True)
plt.rcParams['figure.figsize'] = [8,5]
plt.rcParams['font.size'] =14
```

```
plt.rcParams['font.weight']= 'bold'
plt.style.use('seaborn-whitegrid')
data = pd.read_csv('data\\kc_house_data.csv')
f = plt.figure(figsize=(14,6))
ax = f.add_subplot(121)
sns.scatterplot(x='grade',y='price',data=data,palette='Set1',ax=ax)
ax.set_title('Scatter plot of grade vs price')

ax = f.add_subplot(122)
sns.scatterplot(x='sqft_living',y='price',data=data,palette='viridis')
ax.set_title('Scatter plot of sqft_living vs price')
plt.savefig('sc.png');
```



In [55]:

```
#Cleaning data for prediction
data=data_pred.corr().abs().stack().reset_index().sort_values(0, ascending=Fals
# combine the variable name columns (Which were only named level_0 and level_1
data['pairs'] = list(zip(data.level_0, data.level_1))
# set index to pairs
data.set_index(['pairs'], inplace = True)
#drop level columns
data.drop(columns=['level_1', 'level_0'], inplace = True)
# renamed correlation column to cc
data.columns = ['cc']
# drop duplicates.
data.drop_duplicates(inplace=True)
```

In [56]:

```
#quick preview to confirm
data[(data.cc>=.70) & (data.cc <1)]
```

Out[56]:

pairs	cc
(grade, sqft_living)	0.76
(sqft_living, bathrooms)	0.76
(sqft_living, price)	0.70

In [ ]:

In [ ]:

In [ ]:

In [ ]: