In [1]:

```
# Importing necessary libraries
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

In [10]:

```
df = pd.read_csv(r"C:\Users\hp\Documents\Wine store data sheet.csv")
df.head()
```

Out[10]:

	winery	wine	year	rating	num_reviews	country	region	price	type	body	acidity
0	Teso La Monja	Tinto	2013	4.9	58	Espana	Toro	995.00	Toro Red	5.0	3.0
1	Artadi	Vina El Pison	2018	4.9	31	Espana	Vino de Espana	313.50	Tempranillo	4.0	2.0
2	Vega Sicilia	Unico	2009	4.8	1793	Espana	Ribera del Duero	324.95	Ribera Del Duero Red	5.0	3.0
3	Vega Sicilia	Unico	1999	4.8	1705	Espana	Ribera del Duero	692.96	Ribera Del Duero Red	5.0	3.0
4	Vega Sicilia	Unico	1996	4.8	1309	Espana	Ribera del Duero	778.06	Ribera Del Duero Red	5.0	3.0

In [11]:

```
#Data Cleaning and validation
df.isnull().sum()
```

Out[11]:

```
winery            0
wine              0
year              2
rating            0
num_reviews       0
country           0
region            0
price             0
type            545
body           1169
acidity        1169
dtype: int64
```

In [12]:

```
#Droping the null values on our dataset
df = df.dropna()
```

In [13]:

```
df.shape
```

Out[13]:

```
(6329, 11)
```

In [17]:

```
df.describe()
```

Out[17]:

	rating	num_reviews	price	body	acidity
count	6329.000000	6329.000000	6329.000000	6329.000000	6329.000000
mean	4.259425	442.292463	65.659082	4.158319	2.946753
std	0.124306	718.597235	162.599997	0.583345	0.247955
min	4.200000	25.000000	4.990000	2.000000	1.000000
25%	4.200000	388.000000	19.980000	4.000000	3.000000
50%	4.200000	402.000000	29.150000	4.000000	3.000000
75%	4.200000	415.000000	60.950000	5.000000	3.000000
max	4.900000	32624.000000	3119.080000	5.000000	3.000000

In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6329 entries, 0 to 7499
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   winery       6329 non-null   object 
 1   wine         6329 non-null   object 
 2   year         6329 non-null   object 
 3   rating       6329 non-null   float64
 4   num_reviews  6329 non-null   int64  
 5   country      6329 non-null   object 
 6   region       6329 non-null   object 
 7   price        6329 non-null   float64
 8   type         6329 non-null   object 
 9   body         6329 non-null   float64
 10  acidity      6329 non-null   float64
dtypes: float64(4), int64(1), object(6)
memory usage: 593.3+ KB
```

In [19]:

```
#I'M REMOVING THE YEAR  BECAUSE IT CONTAIN ALOT OF NV MAKING IT DIFFICUT TO CONVER
df['year'] = df['year'].replace('N.V.', np.NaN)
df = df.dropna()
df['year'] = df['year'].astype(np.int64)
```

In [20]:

```
#The column country only has one value so it would not be helpful at all for our mo
df = df.drop(columns=['country'])
df.head()
```

Out[20]:

	winery	wine	year	rating	num_reviews	region	price	type	body	acidity
0	Teso La Monja	Tinto	2013	4.9	58	Toro	995.00	Toro Red	5.0	3.0
1	Artadi	Vina El Pison	2018	4.9	31	Vino de Espana	313.50	Tempranillo	4.0	2.0
2	Vega Sicilia	Unico	2009	4.8	1793	Ribera del Duero	324.95	Ribera Del Duero Red	5.0	3.0
3	Vega Sicilia	Unico	1999	4.8	1705	Ribera del Duero	692.96	Ribera Del Duero Red	5.0	3.0
4	Vega Sicilia	Unico	1996	4.8	1309	Ribera del Duero	778.06	Ribera Del Duero Red	5.0	3.0

In [ ]:

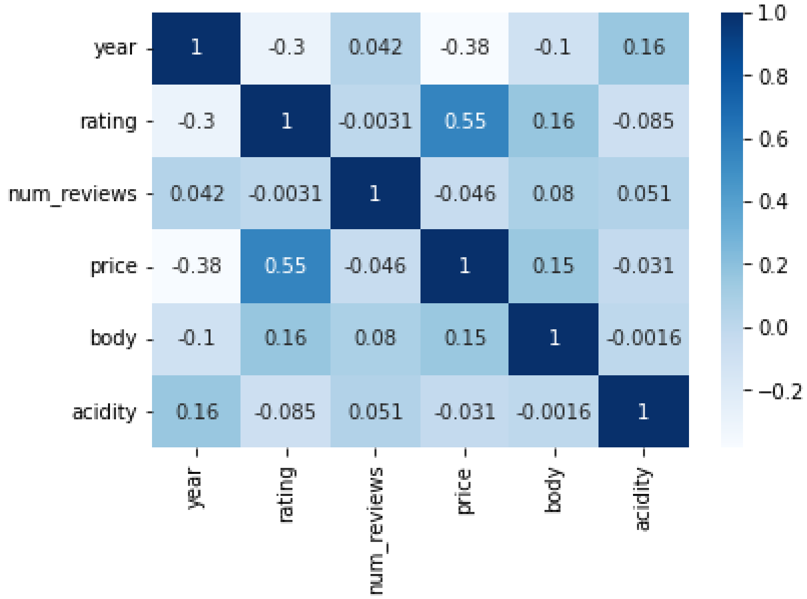
```
#OUR DATA IS CLEAN AND READY FOR Exploratory Data Anaysis
```

In [21]:

```
sns.heatmap(df.corr(), annot=True, cmap='Blues')
```

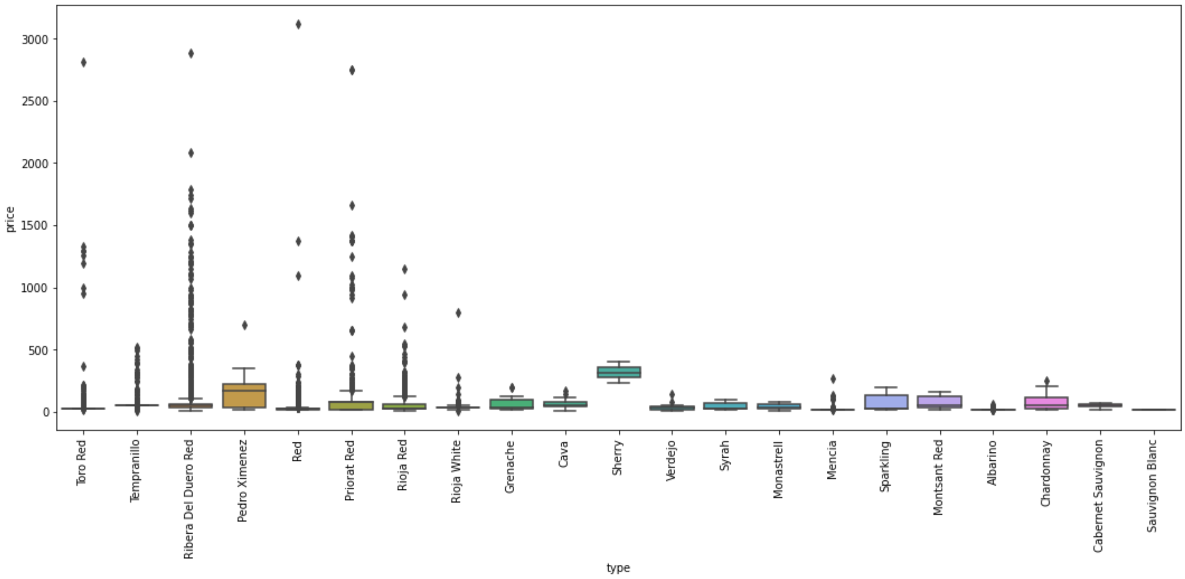
Out[21]:

```
<AxesSubplot:>
```



In [22]:

```
#WE WANT TO CHECK IF THE WINE AFFECTS THE WINE PRICE
fig, ax = plt.subplots(ncols=1, figsize=(18,7))
sns.boxplot(y='price', x='type', data=df, ax=ax)
plt.xticks(rotation=90)
plt.show()
```



In [23]:

```python
print('Categorical columns: ')
for col in df.columns:
    if df[col].dtype == 'object':
        print(str(col))
        label = LabelEncoder()
        label = label.fit(df[col])
        df[col] = label.transform(df[col].astype(str))
```

```
Categorical columns:
winery
wine
region
type
```

In [24]:

```python
df = (df-df.mean())/df.std()
df.head()
```

Out[24]:

	winery	wine	year	rating	num_reviews	region	price	type	body
0	1.281015	1.291723	0.006271	5.114070	-0.631438	1.143410	5.604341	2.208328	1.407862
1	-1.766666	1.546400	0.706038	5.114070	-0.676061	1.355532	1.486888	1.932690	-0.275693
2	1.493195	1.374770	-0.553542	4.314853	2.235989	0.294921	1.556066	0.003224	1.407862
3	1.493195	1.374770	-1.953074	4.314853	2.090552	0.294921	3.779491	0.003224	1.407862
4	1.493195	1.374770	-2.372934	4.314853	1.436084	0.294921	4.293644	0.003224	1.407862

In [27]:

```python
#Splitting the data using train test split
X = df.drop(columns=['price'])
y = df['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2)
```

In [28]:

```python
 #import models for regression
from sklearn.linear_model import LinearRegression, Lasso, Ridge, BayesianRidge
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import LinearSVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [29]:

```
models = {}
def train_validate_predict(regressor, x_train, y_train, x_test, y_test, index):
    model = regressor
    model.fit(x_train, y_train)

    y_pred = model.predict(x_test)

    r2 = r2_score(y_test, y_pred)
    models[index] = r2
```

In [42]:

```
model_list = [LinearRegression, Lasso, Ridge, BayesianRidge, DecisionTreeRegressor,
              RandomForestRegressor]
model_names = ['Linear Regression', 'Lasso', 'Ridge', 'Bayesian Ridge', 'Decision 
               'KNeighbors Regressor', 'Random Forest Regressor']
```

In [43]:

```
models
```

Out[43]:

```
{'Linear Regression': 0.4181414658706233,
 'Lasso': -0.0015752423477575217,
 'Ridge': 0.4181350218488735,
 'Bayesian Ridge': 0.41800899693022064,
 'Decision Tree Regressor': 0.2772944875167366,
 'Linear SVR': 0.1950185249605486,
 'KNeighbors Regressor': 0.7190891654351761,
 'Random Forest Regressor': 0.6417788308955521}
```

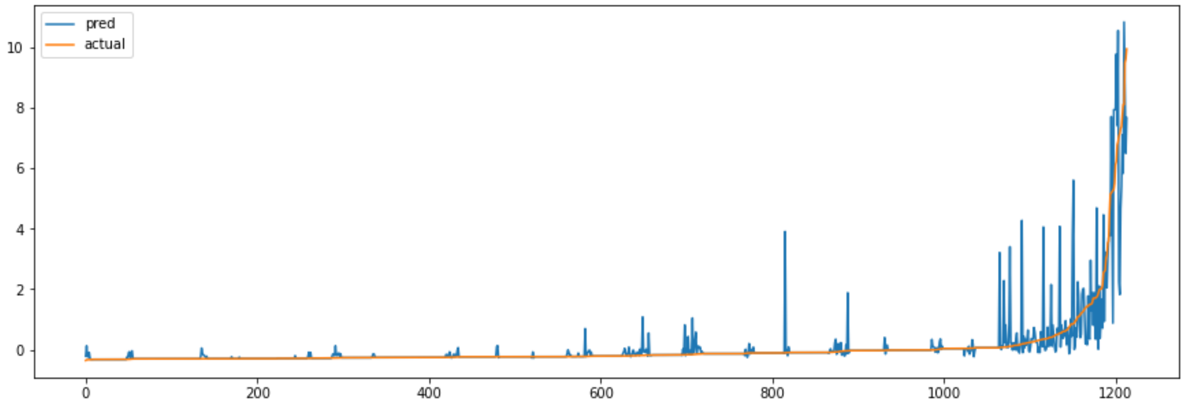
In [33]:

```
 #Evaluating
model = KNeighborsRegressor()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
preds = pd.DataFrame({'y_pred': y_pred, 'y_test':y_test})
preds = preds.sort_values(by='y_test')
preds = preds.reset_index()
```

In [34]:

```
#VISUALIZATION
plt.figure(figsize=(15, 5))
plt.plot(preds['y_pred'], label='pred')
plt.plot(preds['y_test'], label='actual')
plt.legend()
plt.show()
```



In [ ]:

```
# columns has a very little to no relationship toward to the wines prices.

#The model did alright at predicting low prices wines but did terrible at high pri
#I think what caused this from happening according to our EDA earlier, that in our
#theres way more data on low prices wines but theres a little data from the high p
```