SVR

October 14, 2020

Coding Assignment 7

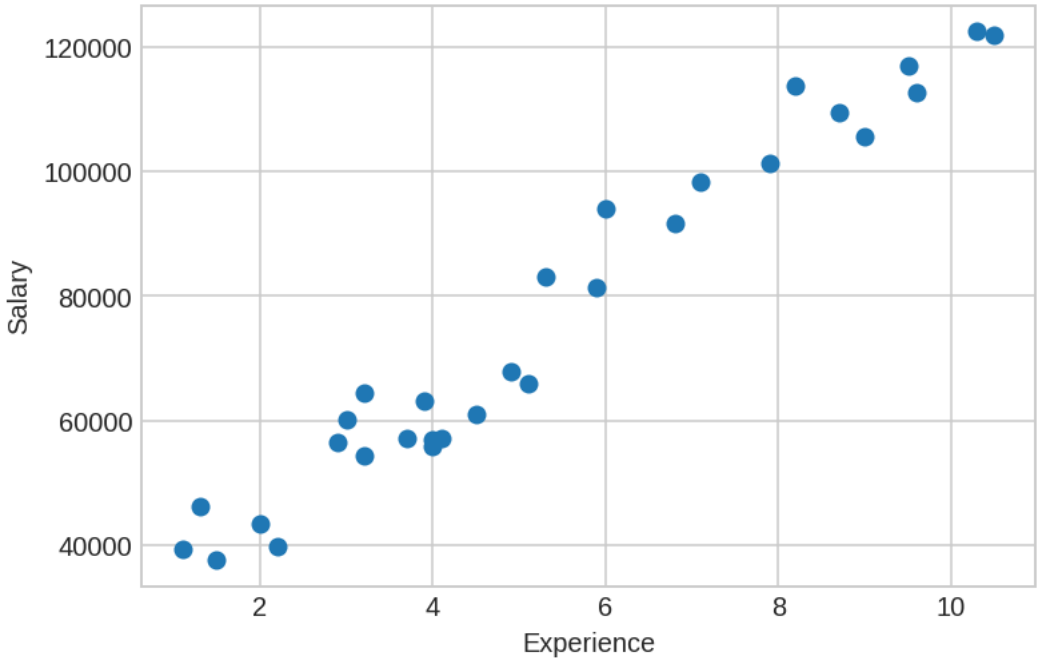
```
SVR (Support Vector Regression
Dataset: Salary vs Experience Dataset
Author: Sreejith S
```

[51]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

%reload_ext autoreload
%matplotlib inline
%autoreload 2
%config InlineBackend.figure_format = 'retina'


#classifiers
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

#evaluation metrics
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

7.1 First, plot the dataset (salary vs experience) using a scatter plot. Put the screenshot of the code, and the scatter plot below.

[52]:
```
#set pd display options
pd.set_option('display.max_columns', 10)
pd.set_option('display.width', 80)

# Importing the dataset
dataset = pd.read_csv('../datasets/salary-experience-dataset.csv')
X = dataset['YearsExperience'].values.reshape(-1, 1)
y = dataset['Salary']#.values.reshape(-1, 1)
```

```
plt.style.use("seaborn-whitegrid")
plt.xlabel("Experience")
plt.ylabel("Salary")
plt.scatter(X, y)
plt.show()
```



7.2 Next, split the dataset into training and testing -keeping a 80-20 split.

[53]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=0)
```

7.3 Build a SVR model (using linear kernel) by fitting the train dataset.

[54]:
```
svr_lin = SVR(kernel='linear', gamma='auto', C=100)
svr = svr_lin.fit(X_train, y_train)
```

7.4 Evaluate the SVR model (using linear kernel) by predicting the values for the test dataset. Report the Mean Absolute Error (MAE) and Mean Squared Error (MSE) for your SVR model.

[55]:
```
print("MAE - test  : ", mean_absolute_error(y_test, svr.predict(X_test)))
print("MAE - train : ", mean_absolute_error(y_train, y_pred = svr.
 ↪predict(X_train)))
print("MSE - test  : ", mean_squared_error(y_test, svr.predict(X_test)))
print("MSE - train : ", mean_squared_error(y_train, y_pred = svr.
 ↪predict(X_train)))
```

```
MAE - test  :  17986.833333333332
MAE - train :  12209.791666666666
MSE - test  :  446168048.1730059
MSE - train :  224889456.54533255
```

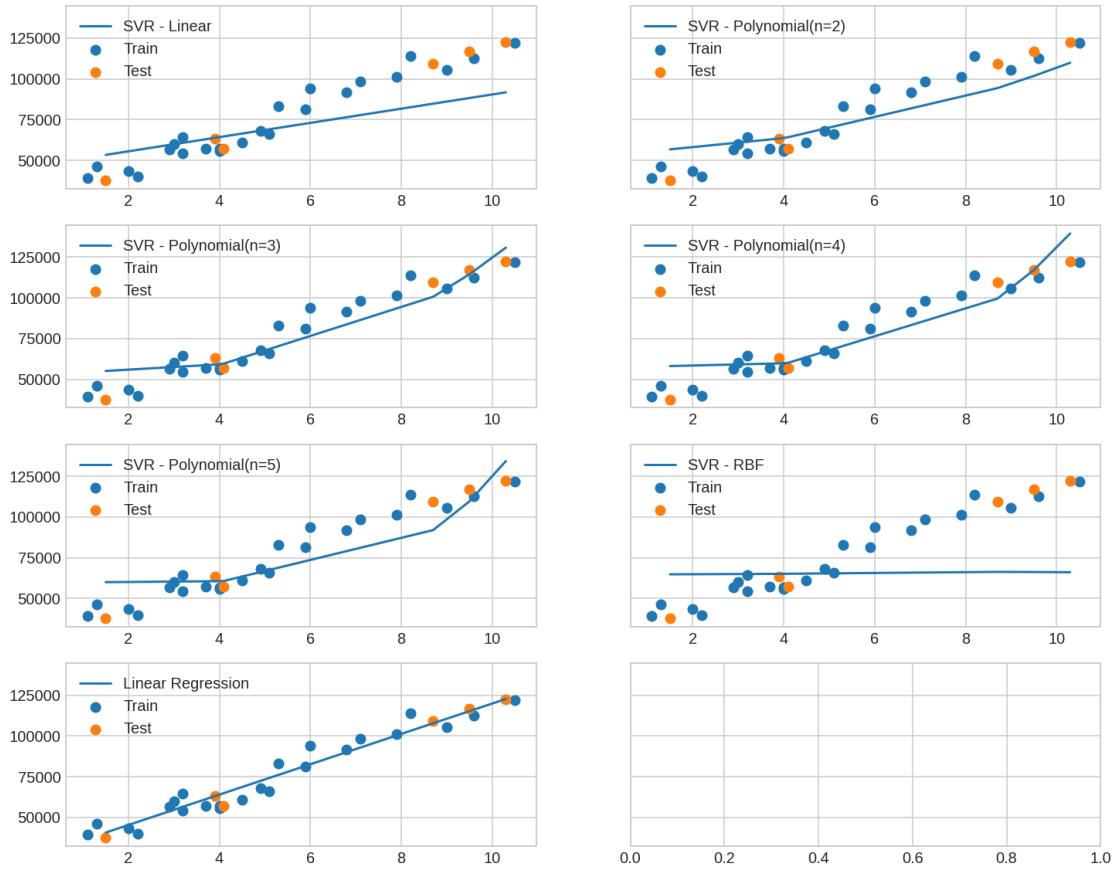
7.5 Build a SVR model (using a polynomial kernel with a few different degrees, and rbf) by fitting the train dataset.

[70]:
```
svr_pol_2 = SVR(kernel='poly', gamma='auto', degree=2)
svr_pol_3 = SVR(kernel='poly', gamma='auto', degree=3)
svr_pol_4 = SVR(kernel='poly', gamma='auto', degree=4)
svr_pol_5 = SVR(kernel='poly', gamma='auto', degree=5)
svr_rbf = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=.1)
lin_reg = LinearRegression()

classifiers = [svr_lin, svr_pol_2, svr_pol_3,
               svr_pol_4, svr_pol_5,
               svr_rbf, lin_reg]
clf_labels = ["SVR - Linear", "SVR - Polynomial(n=2)", "SVR - Polynomial(n=3)",
              "SVR - Polynomial(n=4)", "SVR - Polynomial(n=5)",
              "SVR - RBF", "Linear Regression"]
```

7.6 & 7.7 Evaluate the above polynomial and rbf SVR models by predicting the values for the test dataset. Report the Mean Absolute Error (MAE) and Mean Squared Error (MSE) for your SVR model. And tabulate the performance of each of the models, and interpret which is the best performing kernel. Also, compare the results to the performance of Linear Regression for this same dataset (initial exercises).

[72]:
```
fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(12, 10), sharey=True)
axes = axes.flatten()

eval_metrics = []
for i, clf in enumerate(classifiers):
    model = clf.fit(X_train, y_train)
    errors = [mean_absolute_error(y_test, model.predict(X_test)),
```

```
                mean_squared_error(y_test, model.predict(X_test)),
                mean_absolute_error(y_train, model.predict(X_train)),
                mean_squared_error(y_train, model.predict(X_train))
               ]
    errors = [f"{i:.2f}" for i in errors]
    eval_metrics.append(errors)
    #plotting
    x_s, y_s = zip(*sorted(zip(X_test, model.predict(X_test))))
    axes[i].plot(x_s, y_s, label=clf_labels[i])
    axes[i].scatter(X_train, y_train, label="Train")
    axes[i].scatter(X_test, y_test, label="Test")
    axes[i].legend(loc='upper left')
plt.show()

from utils import plot_table
fig, ax = plt.subplots()
plot_table(eval_metrics, ax, clf_labels, ["MAE Test", "MSE Test", "MAE Train",␣
 ↪"MSE Train"])
plt.show()
```



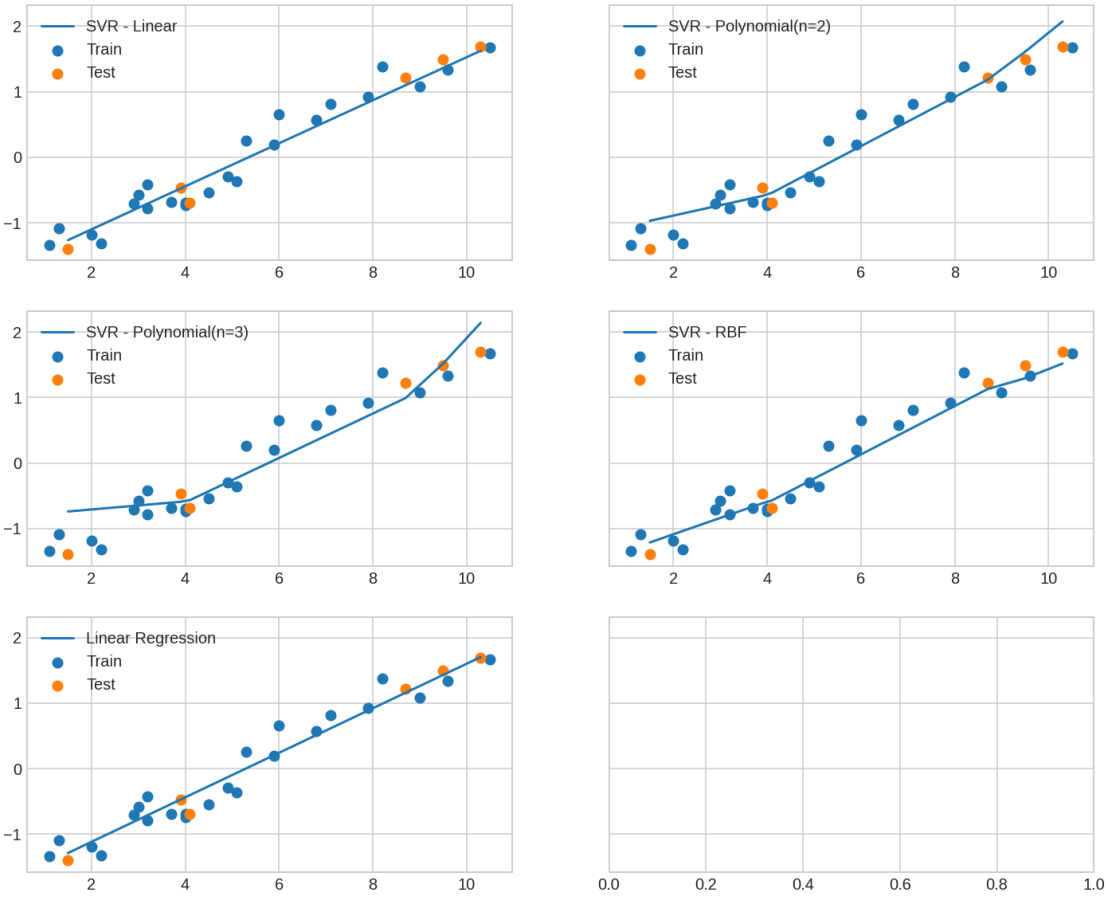
	MAE Test	MSE Test	MAE Train	MSE Train
SVR - Linear	17986.83	446168048.17	12209.79	224889456.55
SVR - Polynomial(n=2)	11523.67	171475351.41	9877.26	135633416.34
SVR - Polynomial(n=3)	7320.92	80479685.53	9047.36	133614379.58
SVR - Polynomial(n=4)	9004.49	138396872.41	11024.94	193610297.60
SVR - Polynomial(n=5)	10891.41	169733307.48	12558.03	251585071.42
SVR - RBF	31203.81	1401370259.46	20371.31	658583072.55
Linear Regression	2446.17	12823412.30	5221.08	36149670.12

Observations

Linear Regression performs the best, followed by SVR with a `polynomial (deg=3)` kernel. This is due to the fact that the linear regression model captures the relations between salary and experience in the best possible way. `rbf` kernal performs poorly, but this is expected as the data is not standardized. We can standardize the data and see the perfomance of `rbf` kernel improve below.

[80]:
```
y_norm  = (y-y.mean())/y.std()
X_train, X_test, y_train, y_test = train_test_split(X, y_norm, test_size=0.2,
                                                    random_state=0)
classifiers = [svr_lin, svr_pol_2, svr_pol_3,
               svr_rbf, lin_reg]
clf_labels = ["SVR - Linear", "SVR - Polynomial(n=2)", "SVR - Polynomial(n=3)",
              "SVR - RBF", "Linear Regression"]

fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(12, 10), sharey=True)
axes = axes.flatten()

eval_metrics = []
for i, clf in enumerate(classifiers):
    model = clf.fit(X_train, y_train)
    errors = [mean_absolute_error(y_test, model.predict(X_test)),
              mean_squared_error(y_test, model.predict(X_test)),
              mean_absolute_error(y_train, model.predict(X_train)),
              mean_squared_error(y_train, model.predict(X_train))
             ]
    errors = [f"{i:.4f}" for i in errors]
    eval_metrics.append(errors)
    #plotting
    x_s, y_s = zip(*sorted(zip(X_test, model.predict(X_test))))
```

```
    axes[i].plot(x_s, y_s, label=clf_labels[i])
    axes[i].scatter(X_train, y_train, label="Train")
    axes[i].scatter(X_test, y_test, label="Test")
    axes[i].legend(loc='upper left')
plt.show()


fig, ax = plt.subplots()
plot_table(eval_metrics, ax, clf_labels, ["MAE Test", "MSE Test", "MAE Train",␣
 ↪"MSE Train"])
plt.show()
```



	MAE Test	MSE Test	MAE Train	MSE Train
SVR - Linear	0.1237	0.0219	0.1875	0.0494
SVR - Polynomial(n=2)	0.2046	0.0627	0.2489	0.0884
SVR - Polynomial(n=3)	0.2669	0.1188	0.3340	0.1709
SVR - RBF	0.1534	0.0250	0.1447	0.0308
Linear Regression	0.0892	0.0171	0.1905	0.0481

[ ]: