SVR

# Coding Assignment 7[¶](#Coding-Assignment-7)

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

In [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.[¶](#X14d3b7482537c8038855f877ee33ee61d4dd2f4)

In [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()

![](data:image/png;base64;base64,)

## 7.2 Next, split the dataset into training and testing -keeping a 80-20 split.[¶](#X8b388460ecdaa4721b4541b6cf117f065ca0c22)

In [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.[¶](#Xced3cd3c772e26ae82b5a58101b9ec2ccf196d6)

In [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.[¶](#X7284cc42016da3d94ee5084cee66465d57425ae)

In [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.[¶](#X01863fb18196723bf25e04276bc71b5dc956f37)

In [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).[¶](#X67b643ed8116837c3bdeae5808d63d6b80436f6)

In [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()

![](data:image/png;base64;base64,)

![](data:image/png;base64;base64,)

### Observations[¶](#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.

In [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()

![](data:image/png;base64;base64,)

![](data:image/png;base64;base64,)

In [ ]: