lab2_solution

October 1, 2019

1 COMP3222/6246 Machine Learning Technologies (2019/20)

1.1 Week 4 – Linear regression, polynomial regression and Support Vector Machines

In the first lab, you implemented a simple linear regression algorithm. Despite its simplicity, this algorithm is quite powerful and can fulfill many of your ML needs. It is not surprising, then, that ML researchers have come up with all sorts of tricks to make it even more effective. In this lab, we have a look at a number of them. However, let us revise the basic model first.

1.2 1. Vanilla linear regression

For the sake of reproducibility, let us set the seed of the random generator:

[1]:
```
import numpy as np

# make numpy randomisation predictable
np.random.seed(0)
```

Then, let us import the Boston house price dataset included in the scikit library:

[2]:
```
from sklearn.datasets import load_boston

# load dataset
boston = load_boston()
```

And split the dataset in two: 80% training set and 20% test set. It sounds familiar, doesn't it?

[13]:
```
# partition the dataset into training and test sets
rnd_indices = np.random.permutation(boston.data.shape[0])
train_size = int(boston.data.shape[0] * 0.8)
train_indices = rnd_indices[:train_size]
test_indices = rnd_indices[train_size:]

train_data = boston.data[train_indices, :]
test_data = boston.data[test_indices, :]
train_target = boston.target[train_indices]
```

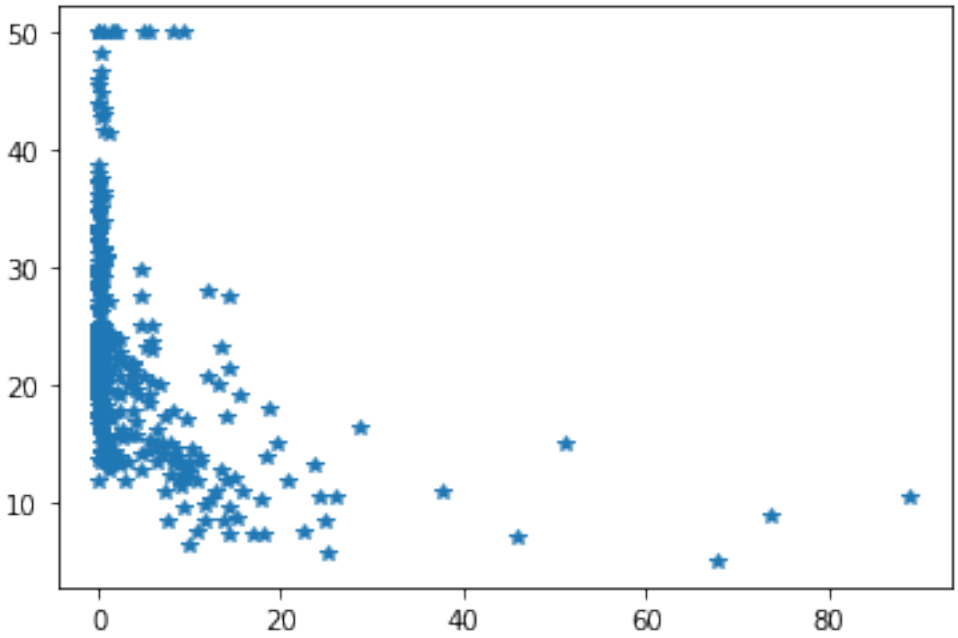
```
test_target = boston.target[test_indices]
```

Exercise 1.1. As a quick warm-up, modify the following code to print the fifth and sixth feature.

[21]:
```
import matplotlib.pyplot as plt

plt.figure()
plt.plot(train_data[:,0], train_target, "*")
```

[21]:
```
[<matplotlib.lines.Line2D at 0x2c08be92c48>]
```



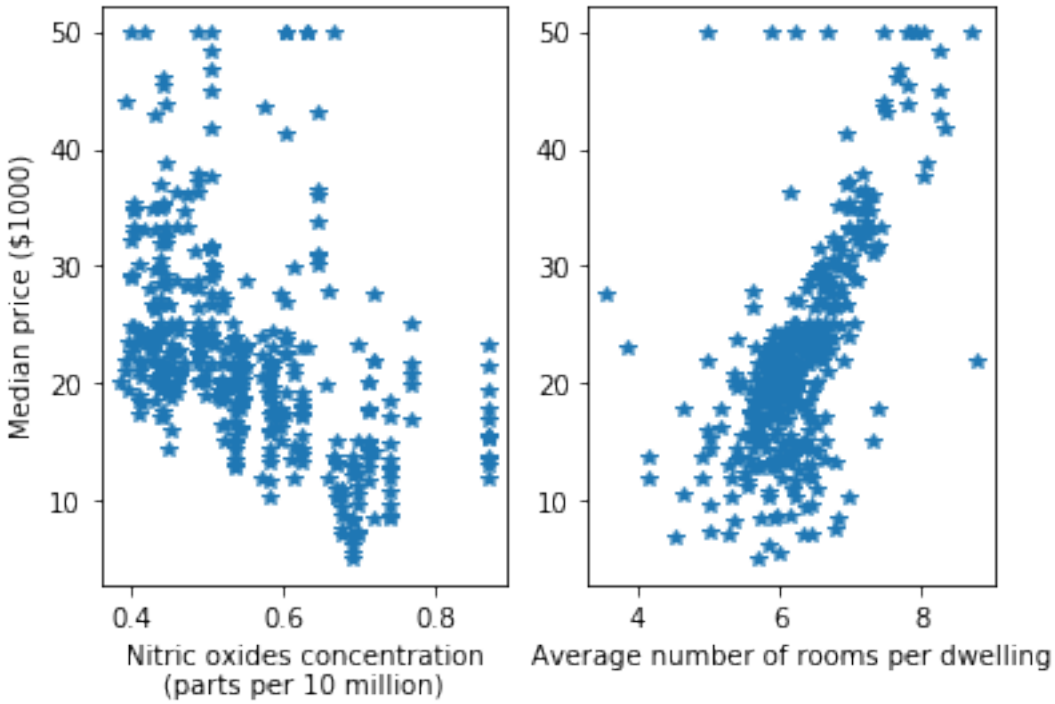
[38]:
```
# Solution

import matplotlib.pyplot as plt

plt.subplot(1,2,1)
plt.plot(train_data[:,4], train_target, "*")
plt.xlabel('Nitric oxides concentration\n(parts per 10 million)')
plt.ylabel('Median price ($1000)')

plt.subplot(1,2,2)
plt.xlabel('Average number of rooms per dwelling')
plt.plot(train_data[:,5], train_target, "*")

plt.show()
```



Finally, it is time to train our linear regression model:

[39]:
```
from sklearn.linear_model import LinearRegression

# fit a linear regressor
lin_reg = LinearRegression()
lin_reg.fit(train_data, train_target)
```

[39]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Exercise 1.2. Modify the following code to print both the training and the testing RMSE.

[40]:
```
train_predict = lin_reg.predict(train_data)
train_rmse = np.sqrt(((train_target - train_predict) ** 2).mean())
print(train_rmse)
```

```
4.854290583096463
```

[47]:
```
# Solution
from sklearn.metrics import mean_squared_error
from math import sqrt

test_predict = lin_reg.predict(test_data)
test_rmse = sqrt(mean_squared_error(test_target, test_predict))
print('Test RMSE = ', test_rmse)
```

```
Test RMSE =  4.060118093883479
```

1.3 2. Feature scaling

The first trick we look at is a data cleaning technique, and quite a general one. Before feeding the training set to your favourite ML algorithm, it is good practice to normalise the input features. This means scaling them so that their values fall more or less in the same range. The scikit library offers two different scaling methods: min-max scaling, and standard scaling.

Exercise 2.1. What do this methods do (explain in 1-2 sentences)? Hint: just google the name of the method and browse the scikit documentation!

Solution Min-max scaling or MinMaxScaler in the Scikit-learn library transforms features by scaling each feature to a given range; see https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxSc
Standard scaling or in the Scikit-learn library transforms features to have zero mean and unit variance; see https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. Have a look at the tutorial https://scikit-learn.org/stable/modules/preprocessing.html since pre-processing is an important step before applying any Machine Learning algorithm.

Let us pick the standard scaling method and write some sample code:

[48]:
```
scaler = StandardScaler()
scaler.fit(train_data)
scaled_train_data = scaler.transform(train_data)
scaled_test_data = scaler.transform(test_data)
```

```
␣
↪---------------------------------------------------------------------------

NameError                                 Traceback (most recent call␣
↪last)

<ipython-input-48-379497088986> in <module>
----> 1 scaler = StandardScaler()
      2 scaler.fit(train_data)
      3 scaled_train_data = scaler.transform(train_data)
      4 scaled_test_data = scaler.transform(test_data)


NameError: name 'StandardScaler' is not defined
```

Exercise 2.2. Modify the code above to print the eleventh feature before and after the transformation.

[49]:
```
# Solution
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(train_data)
scaled_train_data = scaler.transform(train_data)
scaled_test_data = scaler.transform(test_data)
```

Exercise 2.3. Retrain the vanilla linear regressor on the scaled training set and make prediction on the (scaled) test set. Print the RMSE on training and test set.

[60]:
```
# Solution

lin_reg = LinearRegression()
lin_reg.fit(scaled_train_data, train_target)
train_predict = lin_reg.predict(scaled_train_data)
test_predict = lin_reg.predict(scaled_test_data)
train_rmse = np.sqrt(((train_target - train_predict) ** 2).mean())
test_rmse = sqrt(mean_squared_error(test_target, test_predict))
print('Train RMSE = ', train_rmse, ', Test RMSE =', test_rmse)
```

```
Train RMSE =  4.854290583096463 , Test RMSE = 4.06011809388348
```

1.4 3. Regularised linear models

ML algorithms are affected by noise and outliers. On top of that, the model we are using might be too powerful for the problem at hand, and end up overfitting. In order to avoid this, we can constrain the result of our learning effort, and avoid choosing extreme values for the weights of our model. This is called *regularisation* and finds many applications across the ML spectrum.

Here, we look at two alternatives for our linear model: *Ridge* regression and *Lasso* regression.

Coding them in scikit is quite easy. Here is an example:

[64]:
```
from sklearn.linear_model import Ridge, Lasso

# fit a ridge regressor
alpha_ridge = 1
ridge_reg = Ridge(alpha_ridge, solver="cholesky")
ridge_reg.fit(train_data, train_target)

# fit a lasso regressor
alpha_lasso = 1
lasso_reg = Lasso(alpha_lasso)
lasso_reg.fit(train_data, train_target)
```

[64]:
```
Lasso(alpha=1, copy_X=True, fit_intercept=True, max_iter=1000, normalize=False,
      positive=False, precompute=False, random_state=None, selection='cyclic',
```

```
        tol=0.0001, warm_start=False)
```

Exercise 3.1. Compare the training and test RMSE for the Ridge, Lasso and vanilla linear regressors on the Boston house price dataset (no feature scaling).

[73]:
```
# Solution
lin_reg = LinearRegression()
lin_reg.fit(train_data, train_target)

train_lin_predict = lin_reg.predict(train_data)
train_ridge_predict = ridge_reg.predict(train_data)
train_lasso_predict = lasso_reg.predict(train_data)
test_lin_predict = lin_reg.predict(test_data)
test_ridge_predict = ridge_reg.predict(test_data)
test_lasso_predict = lasso_reg.predict(test_data)

train_lin_rmse = sqrt(mean_squared_error(train_target, train_lin_predict))
train_ridge_rmse = sqrt(mean_squared_error(train_target, train_ridge_predict))
train_lasso_rmse = sqrt(mean_squared_error(train_target, train_lasso_predict))
test_lin_rmse = sqrt(mean_squared_error(test_target, test_lin_predict))
test_ridge_rmse = sqrt(mean_squared_error(test_target, test_ridge_predict))
test_lasso_rmse = sqrt(mean_squared_error(test_target, test_lasso_predict))
print('Linear Train RMSE =', train_lin_rmse, ', Linear Test RMSE =',␣
 ↪test_lin_rmse, ', Ridge Train RSME =', train_ridge_rmse, ', Ridge Test RMSE␣
 ↪=', test_ridge_rmse, ', Lasso Train RMSE =', train_lasso_rmse, ', Lasso Test␣
 ↪RMSE =', test_lasso_rmse)
```

```
Linear Train RMSE = 4.854290583096463 , Linear Test RMSE = 4.060118093883479 ,
Ridge Train RSME = 4.874044350170759 , Ridge Test RMSE = 4.115561183052367 ,
Lasso Train RMSE = 5.323133981182158 , Lasso Test RMSE = 5.422054114286459
```

Exercise 3.2. Do the results change much if we scale the input features beforehand (compare the RMSE in both cases)?

[74]:
```
# Solution
lin_reg = LinearRegression()
lin_reg.fit(scaled_train_data, train_target)

train_lin_predict = lin_reg.predict(scaled_train_data)
train_ridge_predict = ridge_reg.predict(scaled_train_data)
train_lasso_predict = lasso_reg.predict(scaled_train_data)
test_lin_predict = lin_reg.predict(scaled_test_data)
test_ridge_predict = ridge_reg.predict(scaled_test_data)
test_lasso_predict = lasso_reg.predict(scaled_test_data)

train_lin_rmse = sqrt(mean_squared_error(train_target, train_lin_predict))
train_ridge_rmse = sqrt(mean_squared_error(train_target, train_ridge_predict))
train_lasso_rmse = sqrt(mean_squared_error(train_target, train_lasso_predict))
```

```
test_lin_rmse = sqrt(mean_squared_error(test_target, test_lin_predict))
test_ridge_rmse = sqrt(mean_squared_error(test_target, test_ridge_predict))
test_lasso_rmse = sqrt(mean_squared_error(test_target, test_lasso_predict))
print('Linear Train RMSE =', train_lin_rmse, 'Linear Test RMSE =',␣
 ↪test_lin_rmse, 'Ridge Train RSME =', train_ridge_rmse, ', Ridge Test RMSE␣
 ↪=', test_ridge_rmse, ', Lasso Train RMSE =', train_lasso_rmse, ', Lasso Test␣
 ↪RMSE =', test_lasso_rmse)
```

```
Linear Train RMSE = 4.854290583096463 Linear Test RMSE = 4.06011809388348 Ridge
Train RSME = 14.532585246027777 , Ridge Test RMSE = 13.90125572860012 , Lasso
Train RMSE = 23.3336715545579 , Lasso Test RMSE = 23.48096387196381
```

Exercise 3.3. What is the best value of alpha_ridge and alpha_lasso (for simplicity use the test set as a validation set)?

[83]:
```
# Solution (obtained by using GridSearch)
from sklearn.model_selection import GridSearchCV

param_grid = {'alpha': [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.
 ↪1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0]}
ridge_reg = Ridge(solver="cholesky")
grid_search = GridSearchCV(ridge_reg, param_grid)
grid_search.fit(train_data, train_target)
grid_search.best_params_
```

```
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\model_selection\_split.py:1978: FutureWarning: The default
value of cv will change from 3 to 5 in version 0.22. Specify it explicitly to
silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
```

[83]:
```
{'alpha': 0}
```

[82]:
```
# Solution (obtained by using GridSearch)
param_grid = {'alpha': [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.
 ↪1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0]}
ridge_reg = Lasso()
grid_search = GridSearchCV(ridge_reg, param_grid)
grid_search.fit(train_data, train_target)
grid_search.best_params_
```

```
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\model_selection\_split.py:1978: FutureWarning: The default
value of cv will change from 3 to 5 in version 0.22. Specify it explicitly to
silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\model_selection\_validation.py:516: UserWarning: With alpha=0,
```

```
this algorithm does not converge well. You are advised to use the
LinearRegression estimator
  estimator.fit(X_train, y_train, **fit_params)
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\linear_model\coordinate_descent.py:475: UserWarning: Coordinate
descent with no regularization may lead to unexpected results and is
discouraged.
  positive)
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\linear_model\coordinate_descent.py:475: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 2830.546216433161, tolerance: 2.112232237918216
  positive)
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\model_selection\_validation.py:516: UserWarning: With alpha=0,
this algorithm does not converge well. You are advised to use the
LinearRegression estimator
  estimator.fit(X_train, y_train, **fit_params)
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\linear_model\coordinate_descent.py:475: UserWarning: Coordinate
descent with no regularization may lead to unexpected results and is
discouraged.
  positive)
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\linear_model\coordinate_descent.py:475: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 3475.801980380252, tolerance: 2.318728802973978
  positive)
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\model_selection\_validation.py:516: UserWarning: With alpha=0,
this algorithm does not converge well. You are advised to use the
LinearRegression estimator
  estimator.fit(X_train, y_train, **fit_params)
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\linear_model\coordinate_descent.py:475: UserWarning: Coordinate
descent with no regularization may lead to unexpected results and is
discouraged.
  positive)
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\linear_model\coordinate_descent.py:475: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 3071.699762324275, tolerance: 2.015264162962963
  positive)
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\model_selection\_search.py:715: UserWarning: With alpha=0, this
algorithm does not converge well. You are advised to use the LinearRegression
estimator
  self.best_estimator_.fit(X, y, **fit_params)
```

```
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\linear_model\coordinate_descent.py:475: UserWarning: Coordinate
descent with no regularization may lead to unexpected results and is
discouraged.
  positive)
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\linear_model\coordinate_descent.py:475: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 4759.955687158083, tolerance: 3.227500324257426
  positive)
```

[82]:
```
{'alpha': 0}
```

1.5 4. Support Vector Machines (for regression)

Support Vector Machines (SVM) can be used not only for classification but also for regression! Furthermore, they already provide an implicit way of regularising the result by changing the width of the margin epsilon.

Exercise 4.1. Modify the code below to train a SVM regressor on the Boston house price dataset. Print the training and test RMSE.

[84]:
```
# fit a support vector machine regressor
epsilon_svm = 1
svm_reg = LinearSVR(epsilon_svm)
svm_reg.fit(train_data, train_target)
```

```
␣
↪---------------------------------------------------------------------------

NameError                                 Traceback (most recent call␣
↪last)

<ipython-input-84-a3227c8a194a> in <module>
  1 # fit a support vector machine regressor
  2 epsilon_svm = 1
----> 3 svm_reg = LinearSVR(epsilon_svm)
  4 svm_reg.fit(train_data, train_target)


NameError: name 'LinearSVR' is not defined
```

[86]:
```
# Solution
from sklearn.svm import LinearSVR
```

```
epsilon_svm = 1
svm_reg = LinearSVR(epsilon_svm)
svm_reg.fit(train_data, train_target)
```

```
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
```

[86]:
```
LinearSVR(C=1.0, dual=True, epsilon=1, fit_intercept=True,
          intercept_scaling=1.0, loss='epsilon_insensitive', max_iter=1000,
          random_state=None, tol=0.0001, verbose=0)
```

Exercise 4.2. What value of epsilon gives you the best performance (for simplicity use the test set as a validation set)?

[89]:
```
# Solution (obtained by using GridSearch)
param_grid = {'epsilon': [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0,␣
 ↪1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0]}
svm_reg = LinearSVR(max_iter=100000)
grid_search = GridSearchCV(svm_reg, param_grid)
grid_search.fit(train_data, train_target)
grid_search.best_params_
```

```
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\model_selection\_split.py:1978: FutureWarning: The default
value of cv will change from 3 to 5 in version 0.22. Specify it explicitly to
silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
```

```
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
```

```
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
```

```
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
```

```
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
```

```
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\model_selection\_search.py:814: DeprecationWarning: The default
of the `iid` parameter will change from True to False in version 0.22 and will
be removed in 0.24. This will change numeric results when test-set sizes are
unequal.
  DeprecationWarning)
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:929:
```

```
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
```

[89]:
```
{'epsilon': 0.3}
```

1.6 5. Polynomial regression

Linear regression is all well and good, but sometimes the dataset requires a non-linear model. In this regard, the ML literature offers quite a range of non-linear regression algorithm. Here we look at the simplest one, *polynomial* regression.

Before, implementing the algorithm, let us create a synthetic dataset:

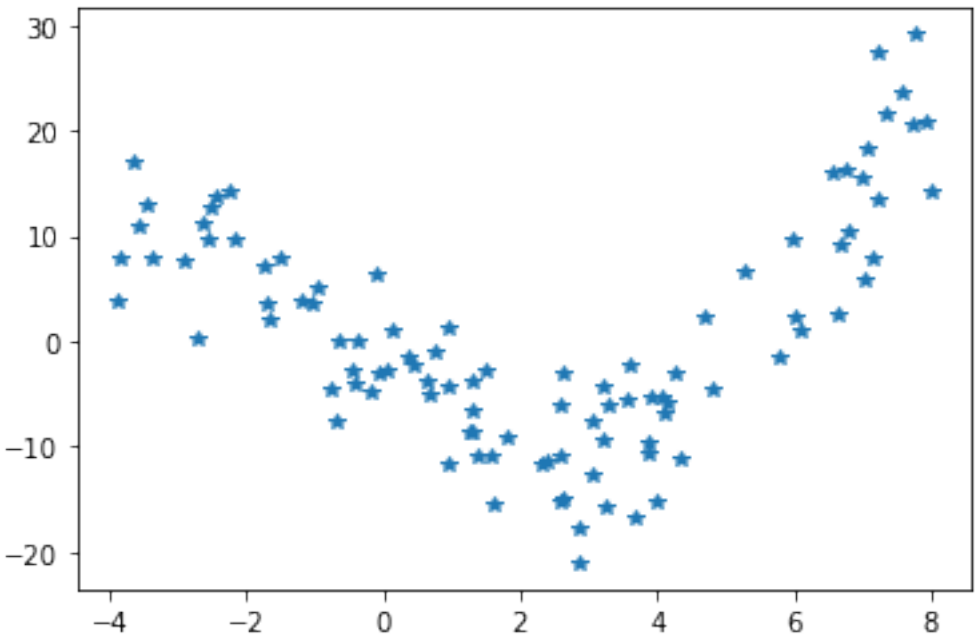
[90]:
```
n = 100
data = 12 * np.random.rand(n, 1) - 3.9
target = 0.09 * (data**3) + 0.3 * (data**2) - 4.1 * data - 2.4 + 4.79 * np.
↪random.randn(n, 1)
```

Exercise 5.1. Plot the dataset.

[91]:
```
# Solution
import matplotlib.pyplot as plt

plt.figure()
plt.plot(data, target, "*")
```

[91]:
```
[<matplotlib.lines.Line2D at 0x2c08e7a8488>]
```



Exercise 5.2. Why would a linear model be a poor choice in this case (explain in 1-2 sentences)?

Solution Clearly, there is no straigtht line that can capture the trend of this generated data.

The idea behind polynomial regression is to expand the number of input features. We do so by taking the existing ones, and multiplying them by one another. More formally, we create an arbitrary number of polynomials of the given input features.

Fortunately, the scikit package allows us to implement this in a few lines of code:

[96]:
```
# fit a quadratic regressor
from sklearn.preprocessing import PolynomialFeatures

poly_features = PolynomialFeatures(degree=2, include_bias=False)
poly_train_data = poly_features.fit_transform(train_data)
lin_reg = LinearRegression()
lin_reg.fit(poly_train_data, train_target)
```

[96]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

The code above creates *quadratic* features, i.e. polynomials of degree two. If we need an even more powerful model, we can choose higher degrees. Of course, this will create an increasingly large number of extra features, and exposes us to overfitting. However, sometimes it is worth the effort.

Exercise 5.3. Which line of the code above you need to change to implement cubic regression (report the modified line)?

Solution the 4th line (changing from degree=2 to degree=3)

Exercise 5.4 Train both Ridge and Lasso regressions and compare the training and test RMSE with the vanilla linear regression shown above.

[98]:
```
# Solution
from sklearn.linear_model import Ridge, Lasso

ridge_reg = Ridge(alpha=1, solver="cholesky")
lasso_reg = Lasso(alpha=1)
ridge_reg.fit(poly_train_data, train_target)
lasso_reg.fit(poly_train_data, train_target)

poly_test_data = poly_features.fit_transform(test_data)

train_lin_predict = lin_reg.predict(poly_train_data)
train_ridge_predict = ridge_reg.predict(poly_train_data)
train_lasso_predict = lasso_reg.predict(poly_train_data)
test_lin_predict = lin_reg.predict(poly_test_data)
test_ridge_predict = ridge_reg.predict(poly_test_data)
test_lasso_predict = lasso_reg.predict(poly_test_data)
```

```
train_lin_rmse = sqrt(mean_squared_error(train_target, train_lin_predict))
train_ridge_rmse = sqrt(mean_squared_error(train_target, train_ridge_predict))
train_lasso_rmse = sqrt(mean_squared_error(train_target, train_lasso_predict))
test_lin_rmse = sqrt(mean_squared_error(test_target, test_lin_predict))
test_ridge_rmse = sqrt(mean_squared_error(test_target, test_ridge_predict))
test_lasso_rmse = sqrt(mean_squared_error(test_target, test_lasso_predict))
print('Linear Train RMSE =', train_lin_rmse, ', Linear Test RMSE =',␣
 ↪test_lin_rmse, ', Ridge Train RSME =', train_ridge_rmse, ', Ridge Test RMSE␣
 ↪=', test_ridge_rmse, ', Lasso Train RMSE =', train_lasso_rmse, ', Lasso Test␣
 ↪RMSE =', test_lasso_rmse)
```

```
Linear Train RMSE = 2.514796223198793 , Linear Test RMSE = 2.7608425106978762 ,
Ridge Train RSME = 2.5983847822856574 , Ridge Test RMSE = 2.8690786754723976 ,
Lasso Train RMSE = 3.307673424475558 , Lasso Test RMSE = 2.8649334890180915
```

```
C:\Local\anaconda3\envs\MLTech\lib\site-
packages\sklearn\linear_model\coordinate_descent.py:475: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 2589.8021978200195, tolerance: 3.227500324257426
  positive)
```

Finally, we can modify our SVM code to do polynomial regression. Notice that the SVM takes as input the regular features and manipulates them with a polynomial kernel:

[100]:
```
from sklearn.svm import SVR

svm_reg = SVR(kernel="poly", degree=2, C=100, epsilon=0.1)
svm_reg.fit(train_data, np.ravel(train_target))
```

```
C:\Local\anaconda3\envs\MLTech\lib\site-packages\sklearn\svm\base.py:193:
FutureWarning: The default value of gamma will change from 'auto' to 'scale' in
version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

[100]:
```
SVR(C=100, cache_size=200, coef0=0.0, degree=2, epsilon=0.1,
    gamma='auto_deprecated', kernel='poly', max_iter=-1, shrinking=True,
    tol=0.001, verbose=False)
```

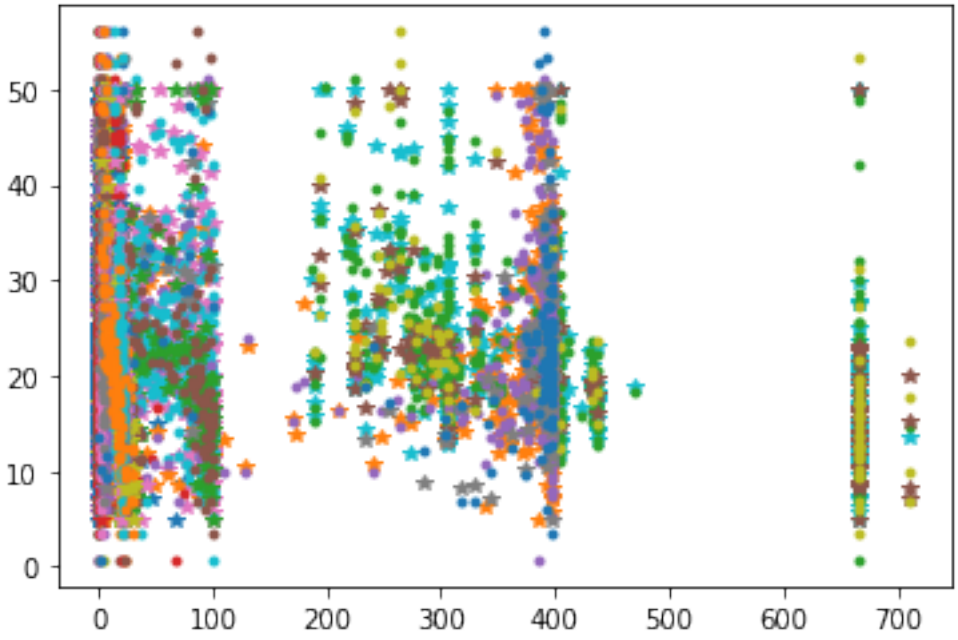
Exercise 5.5 Plot the original dataset as in Exercise 5.1, but add the predictions of the four models (vanilla linear, Ridge, Lasso and SVM). Hint: you can modify the code below and print four separate plots.

[109]:
```
# Solution
train_rmse = sqrt(mean_squared_error(train_target, train_lin_predict))
test_rmse = sqrt(mean_squared_error(test_target, test_lin_predict))

plt.figure()
```

```
plt.plot(train_data, train_target, "*", train_data, train_lin_predict, ".",␣
 ↪test_data, test_target, "*", test_data, test_lin_predict, ".")
print('Train RMSE =', train_rmse, ', Test RMSE =', test_rmse)
```

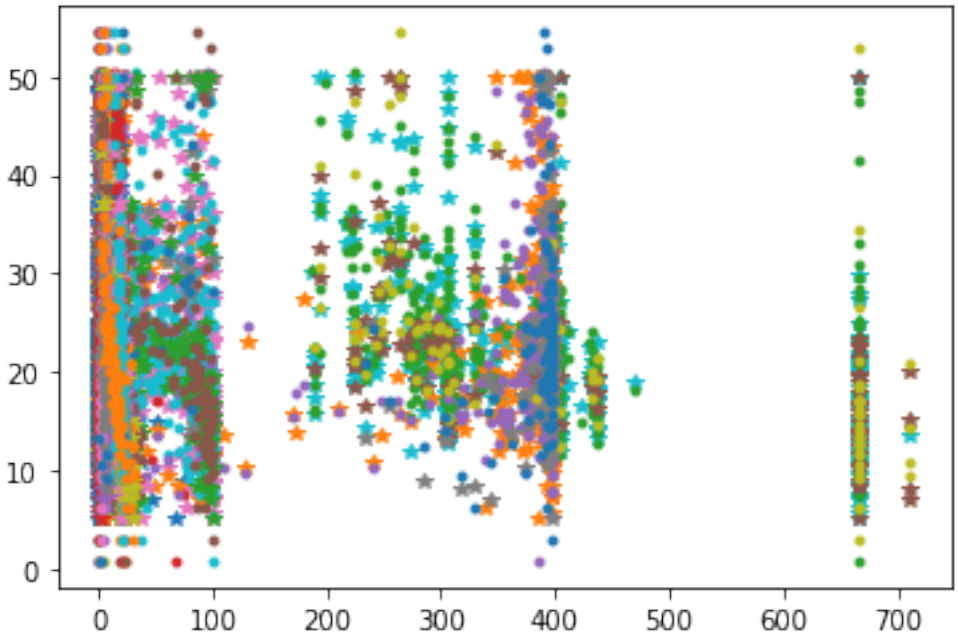
```
Train RMSE = 2.514796223198793 , Test RMSE = 2.7608425106978762
```



[110]:
```
# Solution
train_rmse = sqrt(mean_squared_error(train_target, train_ridge_predict))
test_rmse = sqrt(mean_squared_error(test_target, test_ridge_predict))

plt.figure()
plt.plot(train_data, train_target, "*", train_data, train_ridge_predict, ".",␣
 ↪test_data, test_target, "*", test_data, test_ridge_predict, ".")
print('Train RMSE =', train_rmse, ', Test RMSE =', test_rmse)
```

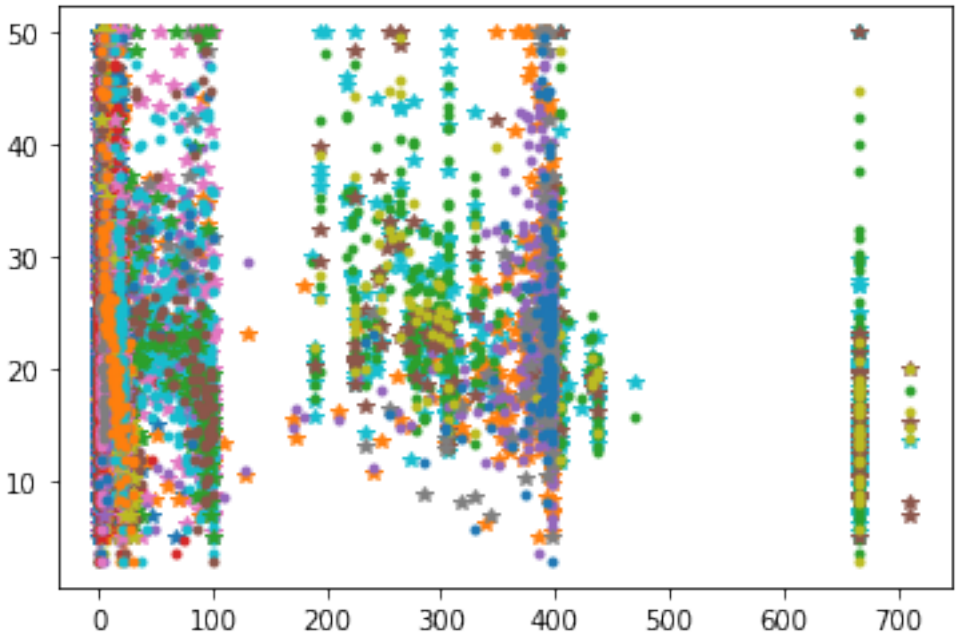
```
Train RMSE = 2.5983847822856574 , Test RMSE = 2.8690786754723976
```



[111]:
```
# Solution
train_rmse = sqrt(mean_squared_error(train_target, train_lasso_predict))
test_rmse = sqrt(mean_squared_error(test_target, test_lasso_predict))

plt.figure()
plt.plot(train_data, train_target, "*", train_data, train_lasso_predict, ".",␣
 ↪test_data, test_target, "*", test_data, test_lasso_predict, ".")
print('Train RMSE =', train_rmse, ', Test RMSE =', test_rmse)
```

```
Train RMSE = 3.307673424475558 , Test RMSE = 2.8649334890180915
```

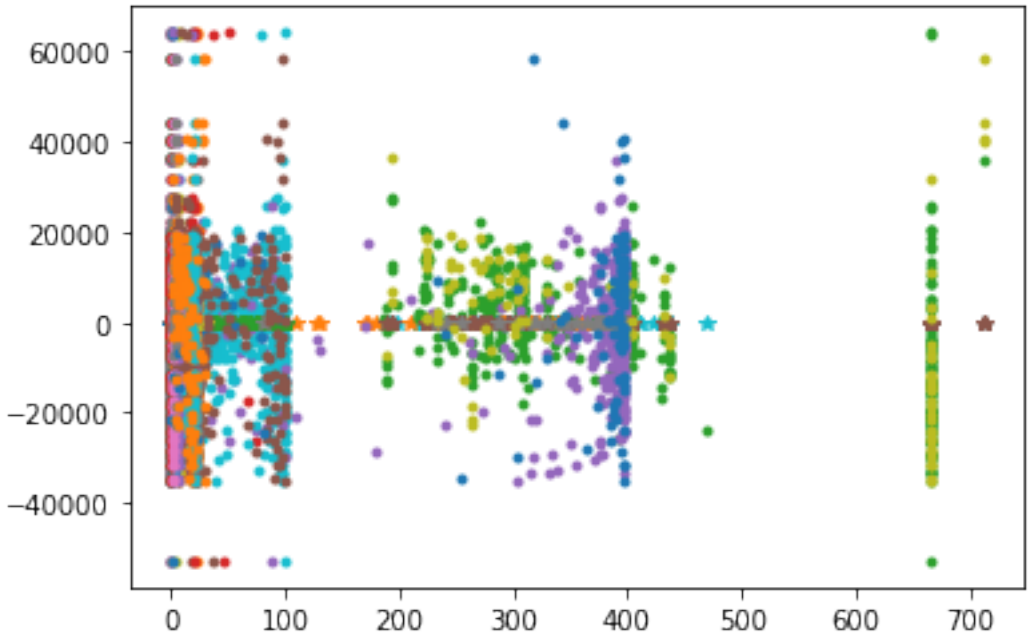


[112]:
```
# Solution
train_svm_predict = svm_reg.predict(train_data)
test_svm_predict = svm_reg.predict(test_data)

train_rmse = sqrt(mean_squared_error(train_target, train_svm_predict))
test_rmse = sqrt(mean_squared_error(test_target, test_svm_predict))

plt.figure()
plt.plot(train_data, train_target, "*", train_data, train_svm_predict, ".",␣
 ↪test_data, test_target, "*", test_data, test_svm_predict, ".")
print('Train RMSE =', train_rmse, ', Test RMSE =', test_rmse)
```

```
Train RMSE = 13313.619161998442 , Test RMSE = 17518.5422842559
```


60000
40000
20000
0
−20000
−40000
0
100
200
300
400
500
600
700