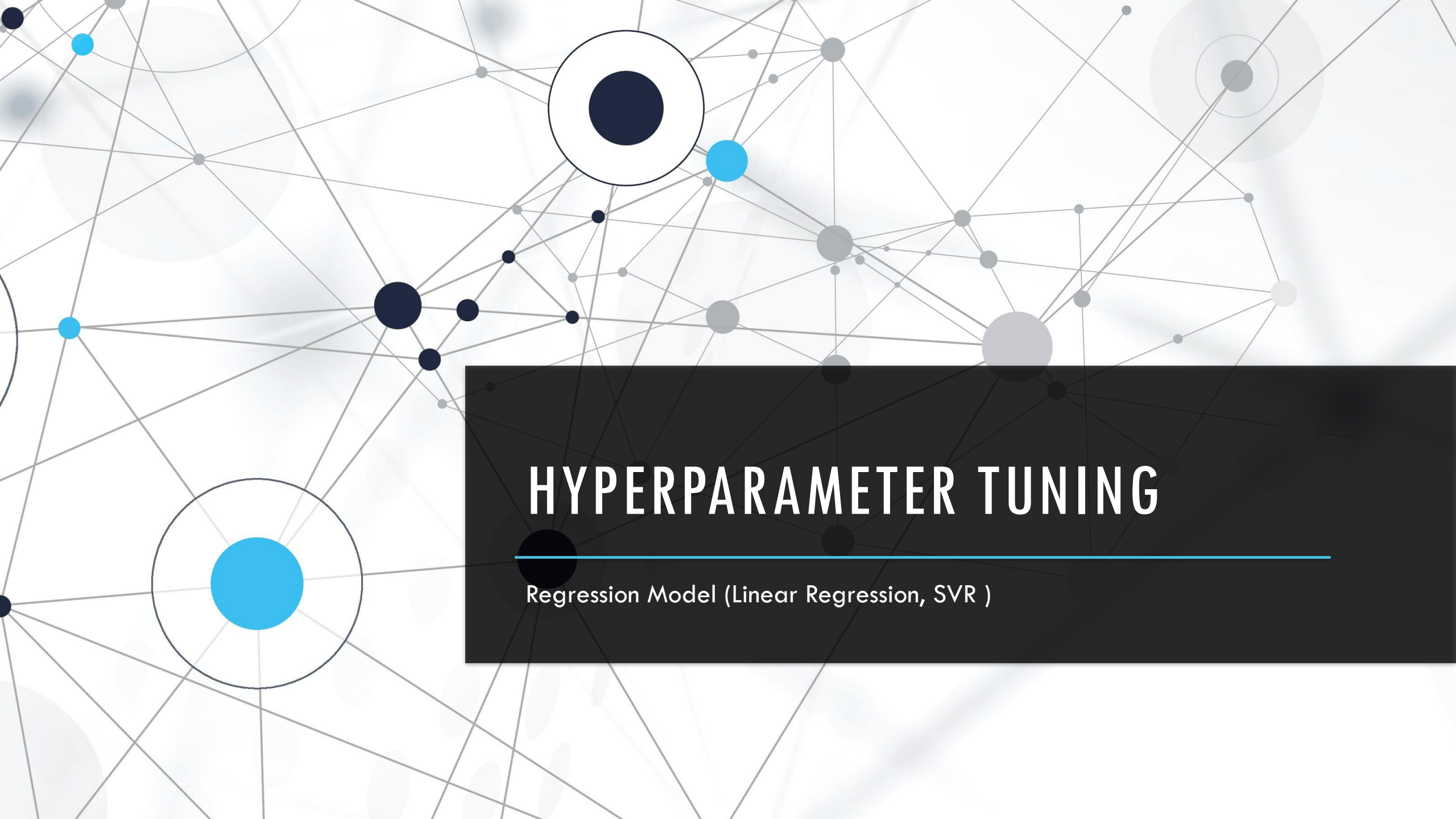


HYPERPARAMETER TUNING

Regression Model (Linear Regression, SVR)

Introduction

01

6.1

Data Preparation

- Data Exploration
- Transform
- Feature Selection
- Train-Validation- Test

02

6.2

Prepare Model, Parameter Dictionary, GridSearchCV()

03

6.3

Prepare Model, Parameter Dictionary, RandomizedCV()

LIBRARIES

1. import numpy as np
2. import pandas as pd
3. import matplotlib.pyplot as plt
4. import seaborn as sns
5. from sklearn import preprocessing
6. from sklearn import model_selection
7. From sklearn.linear_model import LinearRegression
8. From sklearn.svm import SVR
9. From sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV()
10. From sklearn import metrics
11. Import pandas_datareader.data as web
12. From time import time

6.1 Data Preparation

(Load data / Transformation/ Feature Selection)
(Train_Test_split)
from Activity #5.1 (a) –(c))

5.1 (a) Get Data

1

- #Read stock data use pandas_datareader.data from web
  - # Get Stock Data
  - stk_tickers = ['MSFT', 'IBM', 'GOOGL']
  - ccy_tickers = ['DEXJPUS', 'DEXUSUK']
  - idx_tickers = ['SP500', 'DJIA', 'VIXCLS']
  - stk_data = web.DataReader(stk_tickers, 'yahoo')
  - ccy_data = web.DataReader(ccy_tickers, 'fred')
  - idx_data = web.DataReader(idx_tickers, 'fred')

2

- Select columns
  - Base = stk_data.loc[:, ('Adj Close', 'MSFT')]
  - X1 = stk_data.loc[:, ('Adj Close', ('GOOGL', 'IBM'))]
  - X2 = ccy_data
  - X3 = idx_data

5.1 (b) Standardized Data (เปลี่ยนค่าในขั้น 2)

1

- #Standardized data (X1, X2, X3) with kept index (date)
  - X1 = standard_scaler.fit_transform(X1.values),index = X1.index, columns=X1.columns)

2

- Calculate ความแตกต่างของค่า ราคา 'Adj Close', 'MSFT') ย้อนหลัง backHisotry วัน
  - backHistory = [30, 45, 60, 90, 180, 240] -> ทดลองหยิบ 3 ค่า 3 รูปแบบ เพื่อดูระยะเวลาการดูค่าข้อมูลย้อนหลังหลายๆแบบและเปรียบเทียบ MSE
  - BH1, BH2, BH3 = backHistory[1], backHistory[3], backHistory[4]
  - Y = base. shift(-return_period)
  - X4_BH1 = base.diff(BH1).shift(- BH1)
  - X4_BH2 = base.diff(BH2).shift(- BH2)
  - X4_BH3 = base.diff(BH3).shift(- BH3)
  - X4 = pd.concat([X4_BH1, X4_BH2, X4_BH3], axis=1)
  - X4.columns = ['MSFT_3DT', 'MSFT_6DT', 'MSFT_12DT']
  - X4 = pd.DataFrame(standard_scaler.fit_transform(X4.values), index = X4.index,columns=X4.columns)

3

- # Forming Dataset
  - X = pd.concat([X1, X2, X3, X4], axis=1)
  - dataset = pd.concat([Y, X], axis=1)

5.1 (c) Data Preparation

1
- # Drop NA
  - Dropna()
- # View Statistics
  - Describe()

2
- # Assign X, Y (drop datetime index)
  - Y = dataset (1st column)
  - X = dataset (2nd :last column)

3
- # feature selection (correlation)
  - Calculate correlation between variables for only continuous data columns
    - corr()
    - Reduce Corr() to Lower Matrix
    - Drop columns if |correlation value| > 0.9

4
- # Train / Test Preparation (try 2 Option)
- Option#1
  - Test_size = 0.3 * len(X)
  - Train_size = 0.7 * len(X)
  - X_train, X_test = X[0:train_size], X[train_size:len(X)]
  - Y_train, Y_test = Y[0:train_size], Y[train_size:len(X)]
- Option #2
  - X_train, X_test, y_train, y_test = model_selection.train_test_split(X, Y, test_size=0.3, random_state=seed)

6.2

Prepare Parameter Dictionary and Linear Regression() , SVR()

(GridSearchCV())

6.2 Create Model List and Parameter Dictionary

1

- # Create Model List
  - regression = { 'LR': LinearRegression(), 'SVR': SVR(), }

2

- # Create Parameter Dictionary for Linear Regression
  - fit_intercept = [True, False]
  - normalize = [True, False]
  - params_LR = dict(fit_intercept = fit_intercept, normalize = normalize)

3

- # Create Parameter Dictionary for SVR
  - kernel = ['linear', 'rbf', 'poly']
  - C_list = [10, 100]
  - ep_list = [0.1, 1, 5]
  - gamma = [0.01, 0.1]
  - degree = [2, 3]
  - params_SVR = dict(kernel = kernel, C = C_list, epsilon = ep_list, gamma = gamma, degree = degree)

6.2 GridSearchCV() -> (a)

```
for EST in regression:
    model = regression[EST]
      if (EST == 'LR'):
            params = params_LR
       else:
            params = params_SVR

      grid = GridSearchCV(   estimator=model, n_jobs = 1,
                                 verbose = 10,
                                 cv = k,
                                 scoring = 'neg_mean_squared_error’,
                                 param_grid = params )

      grid_result = grid.fit(X_train, y_train)
```

6.2 GridSearchCV() -> (b)

1

- # Show Best Parameters for both models
  - print('Best params: ',grid_result.best_params_)
  - print('Best score: ', grid_result.best_score_)

2

- # Show Score for each parameter combination for both model

```
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
  print("%f (%f) with: %r" % (mean, stdev, param))
```

3

- # Display Mean, std, params
  - Bar()

6.3

Prepare Parameter Dictionary and LinearRegression() , SVR()

(RandomizedSearchCV())

6.2 Create Model List and Parameter Dictionary

1

- # Create Model List
  - regression = { 'LR': LinearRegression(), 'SVR': SVR(), }

2

- # Create Parameter Dictionary for Linear Regression
  - fit_intercept = [True, False]
  - normalize = [True, False]
  - params_LR = dict(fit_intercept = fit_intercept, normalize = normalize)

3

- # Create Parameter Dictionary for SVR

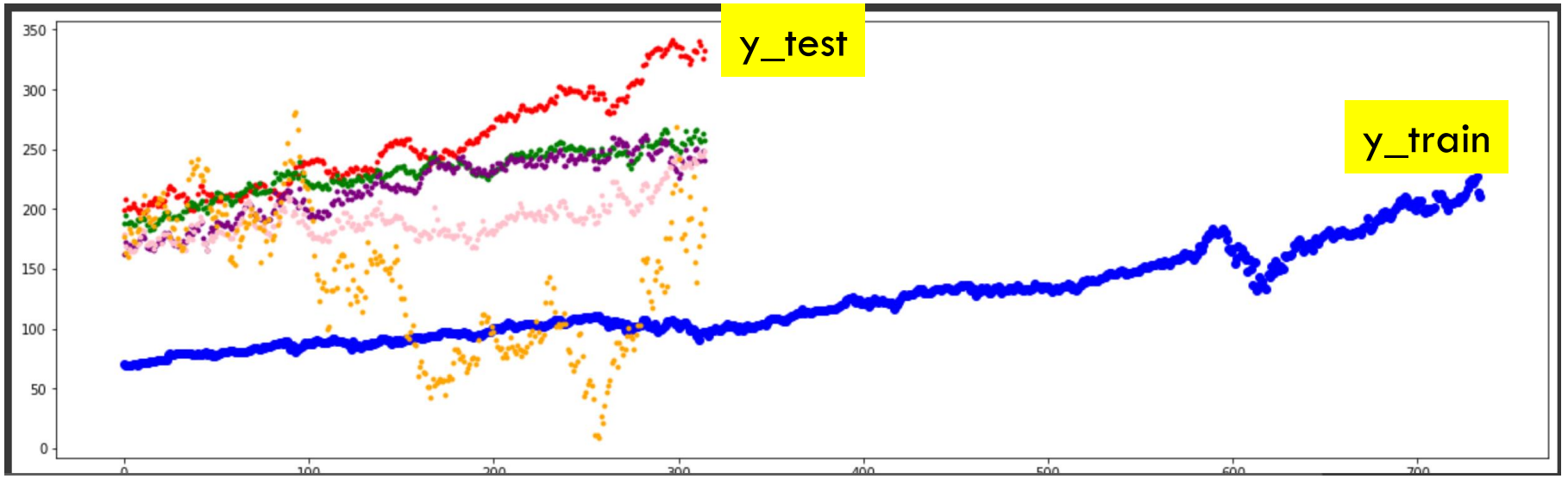
```
kernel = ['linear', 'rbf', 'poly']
C_list = list(np.linspace(0.1, 150, 5, dtype = float))
ep_list = list(np.linspace(0.1, 1, 5, dtype = float))
gamma = list(np.linspace(0.01, 0.1, 5, dtype = float))
degree = [2, 3]

params_SVR = dict( kernel = kernel, C = C_list, epsilon = ep_list, gamma = gamma, degree = degree )
```

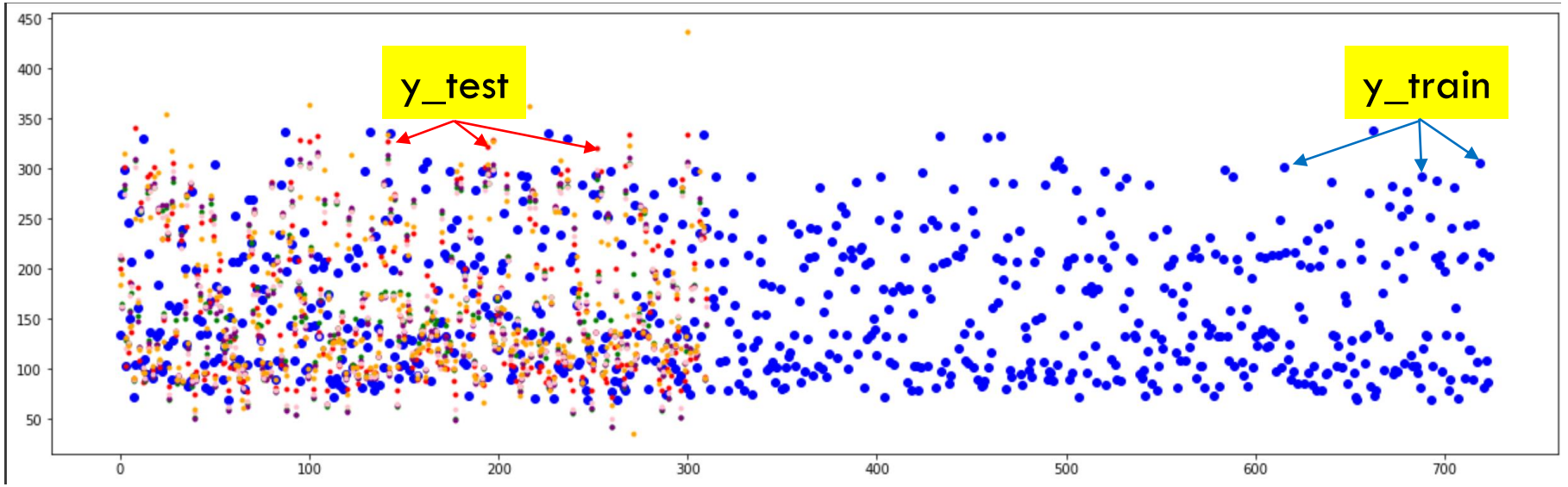
4

```
# Show scatter plot compare y_test vs each model prediction
 plt.scatter();
```

Option#1



Option#2



6.3 RandomizedSearchCV() -> (a)

```
for EST in regression:
    model = regression[EST]
      if (EST == 'LR'):
            params = params_LR
       else:
            params = params_SVR

      grid_rand = RandomizedSearchCV(   estimator=model,   n_jobs = 1,
                                                verbose = 10,
                                                cv = k,
                                                scoring = 'neg_mean_squared_error',
                                                param_distribution = params )

      grid_rand_result = grid_rand.fit(X_train, y_train)
```

6.3 RandomizedCV() -> (b)

1

- # Show Best Parameters for both models
  - print('Best params: ',grid_rand_result.best_params_)
  - print('Best score: ', grid_rand_result.best_score_)

2

- # Show Score for each parameter combination for both model

```
means = grid_rand_result.cv_results_['mean_test_score']
stds = grid_rand_result.cv_results_['std_test_score']
params = grid_rand_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
  print("%f (%f) with: %r" % (mean, stdev, param))
```

3

- # Display Mean, std, params
  - Bar()

4

- # Show scatter plot compare y_test vs each model prediction
- plt.scatter();