In [1]:
```
import sklearn
import numpy as np
```

In [2]:
```
from sklearn import datasets
```

In [3]:
```
housing = datasets.fetch_california_housing()
```

In [4]:
```
housing.feature_names, housing.target_names
```

Out[4]:
```
(['MedInc',
  'HouseAge',
  'AveRooms',
  'AveBedrms',
  'Population',
  'AveOccup',
  'Latitude',
  'Longitude'],
 ['MedHouseVal'])
```

In [5]:
```
housing.data.shape
```

Out[5]:
```
(20640, 8)
```

In [6]:
```
import pandas as pd
df = pd.DataFrame(housing.data, columns=housing.feature_names)
```

In [7]:
```
df.head(3)
```

Out[7]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24

In [8]:
```
import scipy.stats as stats
```

In [9]:
```
# option pricing example
def BS_option_price(S0, K, T, r, q, v):
  d1 = (np.log(S0/K) + (r-q-v**2/2)*T)/(v*np.sqrt(T))
  d2 = d1 - v*np.sqrt(T)
  price = S0*np.exp(-q*T)*stats.norm.cdf(d1) - K*np.exp(-r*T)*stats.norm.cdf(d2)
  return price
```

In [10]:
```
# features = (S0, v, T), target = p
K, T, r, q, v = 180, 0.5, 0.05, 0.015, 0.20
N = 1000
rng = np.random.default_rng(123)
S0 = 180 * rng.uniform(0.6, 1.4, size=N)
v = rng.uniform(0.05, 1.0, size=N)
T = rng.uniform(0.05, 1.0, size=N)
p = BS_option_price(S0, K, T, r, q, v)
```

In [11]:
```
S0 = S0.reshape(N, 1)
v = v.reshape(N, 1)
T = T.reshape(N, 1)
x = np.concatenate((S0, v, T), axis=1)
x.shape, p.shape
```

Out[11]:
```
((1000, 3), (1000,))
```

In [16]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train,  y_test = train_test_split(x, p, test_size=0.20)
```

In [17]:
```
x_train.shape, x_test.shape
```

Out[17]:
```
((800, 3), (200, 3))
```

In [19]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x_train)
X_train = scaler.transform(x_train)
X_test = scaler.transform(x_test)
```

In [20]:
```
X_train[0], x_train[0]
```

Out[20]:
```
(array([ 0.83986966, -1.70389484,  0.47184987]),
 array([2.13886989e+02, 5.85042021e-02, 6.48859774e-01]))
```

In [21]:
```
from sklearn import svm
```

In [23]:
```
model = svm.SVR(gamma=0.01, C=1)
model.fit(X_train, y_train)
```

Out[23]:
```
SVR
SVR(C=1, gamma=0.01)
```

In [26]:
```
y_predict = model.predict(X_test[:5])
```

In [27]:
```
y_predict - y_test[:5]
```

Out[27]:
```
array([ 11.27782881, -29.47726548,   2.5667344 , -24.16911154,
        -8.87236148])
```

In [28]:
```
model.score(X_test, y_test)
```

Out[28]:
```
0.6376575241182911
```

In [29]:
```
# cross validation
from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(model, X_train, y_train, cv=3)
```

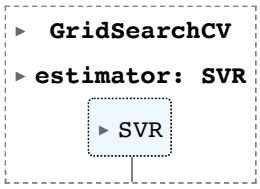
In [31]:
```
cv_scores.mean(), cv_scores
```

Out[31]:
```
(0.48029424472725263, array([0.51491538, 0.45849316, 0.46747419]))
```

In [43]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import TimeSeriesSplit
split = TimeSeriesSplit(n_splits=3)
parm_grid = [
    {'C': [1.0, 10.0], #,100.0]},
    'gamma': [0.01, 0.1]}
]
grid = GridSearchCV(model, param_grid=parm_grid, cv=3, #cv=split
                    return_train_score=True)
```

In [44]:
```
grid.fit(X_train, y_train)
```

Out[44]:



In [45]:
```
best_model = grid.best_estimator_
```

In [46]:
```
grid.best_params_
```

Out[46]:
```
{'C': 10.0, 'gamma': 0.1}
```

In [38]:
```
grid.best_score_
```

Out[38]:
```
0.9396036694332257
```