```
# %%

# ============================================================================
# MODEL SELECTION BY GRID_SEARCH_CV
# ============================================================================


# %%

# importing the library

from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LogisticRegression
# %%

# generating the dataset for 1000 samples with linear relation from make_regression() fucntion of skearn

x, y = datasets.make_regression(n_samples=1000,  # number of samples
                                n_features=20,  # number of features
                                noise=1)  # bias and standard deviation of the guassian noise


# %%

# creating training and test datasets with train_test_split func. of sklearn.model_selection

train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.2)

print("train data size:", len(train_x))  # 800

print("test data size:", len(test_x))  # 200


# %%

# RandomForest Regressor

parameters = {
    "n_estimators": [5, 10, 15, 20, 25],
    "max_depth": [3, 5, 7, 9, 11, 13],
}

model_random_forest = RandomForestRegressor()(
```

```
    class_weight='balanced',
)

model_random_forest = GridSearchCV(
    model_random_forest,
    parameters,
    cv=5,
    scoring='accuracy',
)

model_random_forest.fit(train_x, train_y)

print('-----')
print(f'Best parameters {model_random_forest.best_params_}')
print('Mean cross-validated accuracy score of the best_estimator',
      model_random_forest.best_params_)
print('-----')


# %%

# Logistic Regression

parameters = {
    "C": [0.001, 0.01, 0.1, 1.],
    "penalty": ["l1", "l2"]
}

model_logistic_regression = LogisticRegression(
    class_weight="balanced",
    solver="liblinear",
)

model_logistic_regression = GridSearchCV(
    model_logistic_regression,
    parameters,
    cv=5,
    scoring='accuracy',
)

model_logistic_regression.fit(train_x, train_y)

print('-----')
print(f'Best parameters {model_random_forest.best_params_}')
print('Mean cross-validated accuracy score of the best_estimator',
      model_random_forest.best_params_)
print('-----')
```

```
# %%

# Support Vector Machines (SVM)


parameters = {
    "C": [0.001, 0.01, 0.1, 1.],
    "kernel": ["linear", "poly", "rbf", "sigmoid"],
    "gamma": ["scale", "auto"],
}

model_svc = SVC(
    class_weight="balanced",
    probability=True,
)

model_svc = GridSearchCV(
    model_svc,
    parameters,
    cv=5,
    scoring='accuracy',
)

model_svc.fit(train_x, train_y)

print('-----')
print(f'Best parameters {model_random_forest.best_params_}')
print('Mean cross-validated accuracy score of the best_estimator',
      model_random_forest.best_params_)
print('-----')


# %%

# KNN


parameters = {
    "weights": ["uniform", "distance"],
}

model_k_neighbors = KNeighborsRegressor()(
)

model_k_neighbors = GridSearchCV(
    model_k_neighbors,
    parameters,
```

```
    cv=5,
    scoring='accuracy',
)

model_k_neighbors.fit(train_x, train_y)


print('-----')
print(f'Best parameters {model_random_forest.best_params_}')
print('Mean cross-validated accuracy score of the best_estimator',
      model_random_forest.best_params_)
print('-----')


# %%

# ============================================================================
# THE END
# ============================================================================
```