ISYE 7406 Final Exam

Fall 2022

Derek Cheng (ID: 903651310)

1. Introduction

In this final exam problem, we will be evaluating the performance of various regression models to estimate two response variables for a dataset: the mean and variance of an unknown distribution for X_1 and X_2. For these two response variables, we will attempt to build two different models using X_1 and X_2 to predict each response. To determine these models, we will perform cross-validation and grid search of hyperparameters to select our best models.

2. Data Source

We are provided with two datasets: train and test. The train dataset consists of 10,000 rows for X_1 and X_2 and 200 realizations for each pair resulting in a shape of 10,000 x 202. To create the two response variables, we calculate the mean and variance for each row's 200 realizations resulting in a final shape of 10,000 x 4. We will use the two response variables as labels in training our regression models.

	0	1	2	3	4	5	6	7	8	9	...
0	0.00	0.00	11.627782	15.492258	18.113715	26.651443	35.760375	14.042800	17.693898	22.111201	...
1	0.00	0.01	32.418541	11.764245	12.497371	40.506724	14.571412	24.105663	14.376934	10.176242	...
2	0.00	0.02	22.027616	22.607331	40.110178	26.467103	9.863613	21.785622	10.163916	24.017838	...
3	0.00	0.03	33.721922	19.678209	12.277800	26.020683	10.881366	10.526013	21.427323	18.666008	...
4	0.00	0.04	12.302136	11.913433	9.100761	20.023935	13.930801	14.081861	17.413838	24.911193	...
...	...	...	...	...	...	...	...	...	...	...	...
9995	0.99	0.95	67.484465	70.918129	57.267698	51.074847	56.152244	69.188008	63.500230	69.019970	...
9996	0.99	0.96	42.866619	72.810027	73.988045	60.944896	67.229097	76.587754	49.184458	66.261434	...
9997	0.99	0.97	57.888105	36.745410	60.188074	67.245414	65.365039	58.566381	71.491106	66.641770	...
9998	0.99	0.98	67.423897	62.534617	65.812597	59.364404	49.112736	57.797796	53.310866	64.614495	...
9999	0.99	0.99	60.049120	61.167793	66.214102	61.119529	58.921983	57.345777	57.797926	52.699489	...

10000 rows × 202 columns

	x1	x2	mu	v
0	0.00	0.00	20.315556	97.676165
1	0.00	0.01	21.115456	112.681665
2	0.00	0.02	20.558719	94.542537
3	0.00	0.03	19.164957	81.550269
4	0.00	0.04	20.051926	99.869165
...	...	...	...	...
9995	0.99	0.95	60.997455	127.980617
9996	0.99	0.96	61.999813	100.911845
9997	0.99	0.97	62.093463	80.265974
9998	0.99	0.98	61.876952	53.491256
9999	0.99	0.99	62.785429	23.685434

10000 rows × 4 columns

Figure 1. Raw data of train dataset (left). Transformed train dataset (right).

To get an idea of the relationship between X_1 and X_2 and the mean and variance response variables, we plot each on a 2D plot to visualize if there is any correlation. As we can see below, it appears that between X_1 and mu that the relationship is positively linear, but the relationship between the other variables are more noisy to discern.

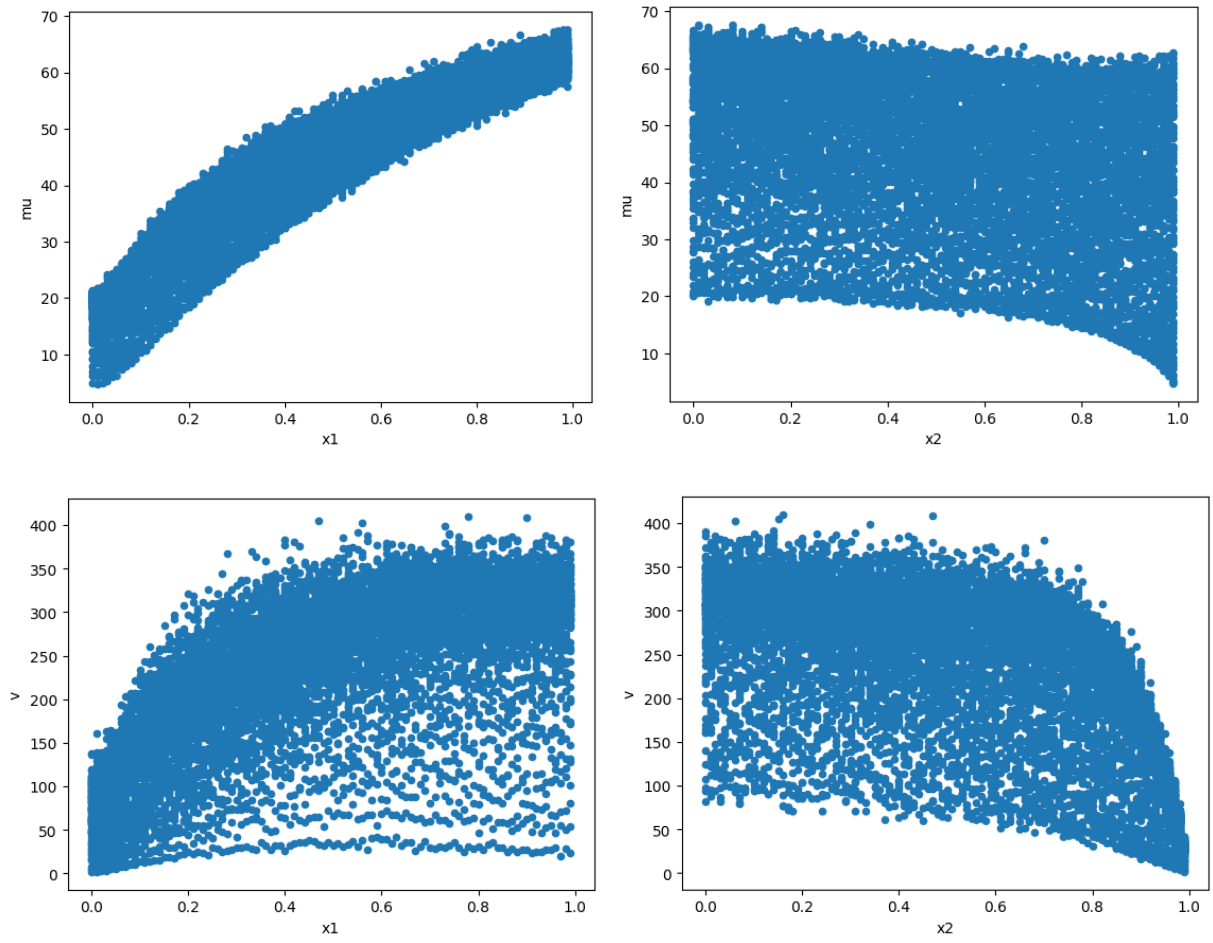

Figure 2. Visualizing relationships between X_1 and X_2 and responses

As for the test dataset, there are 2500 samples of X_1 and X_2 for which we will use our best two models to predict the mean and variance response variables for each pair.

3. Proposed Methodology

First, we will compare performance between 15 different regression models with their default parameters. Then, we will select the top 5 best performing models to perform grid search for best parameters. Afterwards, we can compare for lowest mean squared error to use as our predicting model. The 16 different models are as described:

- Linear Regression - linear model minimizing sum of squares between target and linear approximations
- Ridge Regression - linear regression model adding penalty to all coefficients to reduce multicollinearity

- Lasso Regression - linear regression model adding penalty by setting coefficients to zero to reduce redundant variables
- Elastic Net - linear regression model trained with both Ridge and Lasso penalties
- Bayesian Ridge - linear regression model trained with gamma prior distributions instead of Gaussian
- Stochastic Gradient Descent (SGD) Regressor - linear regression model trained with a gradient descent and learning rate instead of Ordinary Least Squares
- K-Neighbors Regressor - Model based on k-Nearest neighbors where the prediction is interpolated by targets from nearest neighbors
- Decision Tree Regressor - tree based model where splits in data are performed at the most optimum points where squared error is least
- Random Forest Regressor - ensemble bagging method of many regression trees created from random splits of the data
- Gradient Boosting Regressor - ensemble boosting method of many regression trees learned in succession of one another
- AdaBoost Regressor - ensemble boosting method of many regression stumps (trees with only a depth of 1) learned in succession of one another
- Support Vectors (SV) Regressor - Model creating a decision function of the data by maximizing margin depending on various kernels selected
- Cat Boost Regressor - ensemble boosting method of many regression trees with the condition of all nodes of all trees at the same level must test the same predictor with equivalent condition
- XGBoost Regressor - another variant of ensemble boosting tree method with gradient descent optimization
- LGBM Regressor - another variant of gradient boosting tree method where leaves are grown depending on bins of data, instead of data points

Now that we've covered the regression methods that we'll be using, we can compare each one to find the best performance on our dataset.

4. Analysis and Results

First, we attempt to find our best model candidates for predicting the mean response variable mu. We perform K-fold cross validation of 10 folds; that is, setting 9

folds for training and testing on one fold, repeated 10 times for an average score. We will perform this on each regression model described and gather its mean-squared error score and R^2 score.

	Classifier	MSE@10	R2@10	MSE@5	R2@5
13	CatBoostRegressor	-1.220404	0.993462	-1.234366	0.993392
15	LGBM Regressor	-1.258871	0.993257	-1.270108	0.993201
10	Gradient Boosting Regressor	-1.285581	0.993114	-1.298049	0.993052
12	SVR	-1.299760	0.993038	-1.315939	0.992958
14	XGB Regressor	-1.364068	0.992694	-1.394810	0.992533
7	K Neighbors Regressor	-1.431618	0.992334	-1.451900	0.992229
9	Random Forest Regressor	-1.531112	0.991816	-1.546360	0.991692
8	Decision Tree Regressor	-2.567616	0.986240	-2.543721	0.986427
11	Adaboost Regressor	-3.645658	0.980704	-3.637115	0.980717
4	Bayesian Ridge	-9.206467	0.950699	-9.208869	0.950718
0	Linear Regression	-9.206467	0.950699	-9.208869	0.950718
1	Ridge Regression	-9.206777	0.950697	-9.209286	0.950716
5	SGD Regressor	-9.211873	0.950677	-9.209080	0.950718
2	Lasso Regression	-33.221295	0.822144	-33.233566	0.822165
6	Kernel Ridge	-132.522377	0.290050	-132.537635	0.290574
3	ElasticNet	-146.965433	0.213243	-146.972551	0.213455

Figure 3. Models with default parameters and scores for predicting mu

In Figure 3 above, we find our top five models to be CatBoost, LGBM, Gradient Boost, SVR, and XGBoost. We also notice that while these models have lowest MSE, they also have a R^2 score, showing that the models have captured most of the variance within the data. We also compare the scores with a K-fold cross-validation of 5 to see if the results change significantly. In this case, the results do not vary by much from 5 folds to 10, but we do notice that with 10 folds, the errors are lower since there is more data to train on in each iteration. Next, we gather our top five models to perform grid search on each model to find the best model with lowest mean squared error.

	Classifier	Best Params	Best Score@10	Variance@10
3	SVR	{'C': 1000, 'gamma': 'scale', 'kernel': 'rbf'}	-1.190601	0.002613
0	CatBoostRegressor	{'depth': 7, 'iterations': 500, 'l2_leaf_reg':...	-1.202861	0.002898
2	Gradient Boosting Regressor	{'learning_rate': 0.01, 'max_depth': 6, 'n_est...	-1.227864	0.002765
4	XGBRegressor	{'gamma': 0.2, 'learning_rate': 0.05, 'max_dep...	-1.239487	0.002560
1	LGBM Regressor	{'learning_rate': 0.05, 'max_depth': 3, 'n_est...	-1.240071	0.002580

Figure 4. Top 5 models with their parameters and scores for mu

Our best regression model SVR is chosen with the lowest score of **1.19** for mean squared error which we will use for prediction on the test dataset. All models have very similar and low variance signifying that the models have learned well, however it is interesting to note that SVR performed better than the ensemble models. We will now attempt the same process for predicting the variance response variable.

	Classifier	MSE@10	R2@10	MSE@5	R2@5
10	Gradient Boosting Regressor	-540.133732	0.930746	-540.711660	0.930722
13	CatBoostRegressor	-543.577631	0.930310	-546.118634	0.930036
15	LGBM Regressor	-547.382708	0.929826	-548.979007	0.929673
14	XGB Regressor	-594.888904	0.923740	-602.782391	0.922784
7	K Neighbors Regressor	-632.004887	0.919008	-631.657199	0.919099
9	Random Forest Regressor	-670.263883	0.914030	-674.390379	0.913682
11	Adaboost Regressor	-698.900275	0.910009	-704.682079	0.910386
12	SVR	-813.072266	0.895873	-839.030482	0.892576
8	Decision Tree Regressor	-1111.483434	0.857663	-1120.418056	0.856794
5	SGD Regressor	-2172.310280	0.721683	-2173.794431	0.721512
4	Bayesian Ridge	-2172.656345	0.721662	-2172.986921	0.721724
0	Linear Regression	-2172.656362	0.721662	-2172.986952	0.721724
1	Ridge Regression	-2172.664147	0.721661	-2172.994530	0.721723
2	Lasso Regression	-2196.629863	0.718610	-2196.884776	0.718674
3	ElasticNet	-6359.204841	0.185260	-6358.756669	0.185634
6	Kernel Ridge	-9178.665199	-0.176630	-9179.506392	-0.175983

Figure 5. Models with default parameters and scores for predicting variance

From Figure 5, we gather that our top 5 models are Gradient Boost, CatBoost, LGBM, XGB, and K-Nearest Neighbors. We also notice that the top R^2 score is 0.93 compared to the top R^2 score of 0.99 for mu, signifying that it is easier for models to learn the mean response variable compared to the variance response variable as we first saw when we plotted the relationships. We now perform grid search on these five models.

	Classifier	Best Params	Best Score@10	Variance@10
0	CatBoostRegressor	{'depth': 5, 'iterations': 500, 'l2_leaf_reg':...	-530.501788	191.975229
2	Gradient Boosting Regressor	{'learning_rate': 0.01, 'max_depth': 4, 'n_est...	-532.529982	192.042925
1	LGBM Regressor	{'learning_rate': 0.01, 'max_depth': 5, 'n_est...	-536.545952	174.497899
4	XGBRegressor	{'gamma': 0.0, 'learning_rate': 0.05, 'max_dep...	-537.509041	161.833461
3	KNeighborsRegressor	{'n_neighbors': 21, 'weights': 'distance'}	-560.791933	327.721290

Figure 6. Top 5 models with their parameters and scores for variance

Our best model was CatBoost with a score of 530 for mean squared error. However, I decided to select XGBoost as the best model since its MSE of **537** is only different by 7, while its variance of **161** is much lower than CatBoost's 191. This means the XGBoost model would provide more reliable predictions.

5. Conclusion

In our experiment, we have concluded that the best model to predict the mean variable was Support Vector machines (SVR) after hypertuning. SVR had the lowest mean squared error. Repeating this same process, we found XGBoost to be the best model to predict the variance response variable, but while it did not have the highest MSE, the variance with significantly less than the other top four models. It was interesting to see SVR outperform the other ensemble models for mu predictions, but this result could change depending on the random seed chosen or if there is a different train and test split. We also found that the top performing models tended to be ensemble methods, while the lowest were often the linear models. This is most likely due to linear models being unable to capture the complexity of the dataset and ensemble methods are able to make use of multiple models to learn and predict more accurately in comparison. Overall, this was a fun way to demonstrate and learn the differences between the models and showcase ways to select the best models.

Appendix: Python Code

```
# Import libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold, cross_val_score, GridSearchCV

# Load dataset
train_df = pd.read_csv("2022Fall7406train.csv", header=None)
test_df = pd.read_csv("2022Fall7406test.csv", header=None)

X1 = train_df.iloc[:,0]
X2 = train_df.iloc[:,1]
mu = train_df.iloc[:,2:].mean(axis=1)
v = train_df.iloc[:,2:].var(axis=1)
train_df2 = pd.DataFrame({'x1':X1, 'x2':X2, 'mu':mu, 'v': v})

# Plot relationships
train_df2.plot(x='x1',y='mu', kind="scatter")
train_df2.plot(x='x2',y='mu', kind="scatter")
train_df2.plot(x='x1',y='v', kind="scatter")
train_df2.plot(x='x2',y='v', kind="scatter")

# Import regression models
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet,
BayesianRidge, SGDRegressor
from sklearn.kernel_ridge import KernelRidge
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import (RandomForestRegressor, GradientBoostingRegressor,
                              AdaBoostRegressor)
from sklearn.svm import SVR
from catboost import CatBoostRegressor
from xgboost.sklearn import XGBRegressor
from lightgbm import LGBMRegressor

# Find the best model for predicting mu
cv_10 = KFold(10, shuffle=True, random_state=7406)
cv_5 = KFold(5, shuffle=True, random_state=7406)
def score_clf(clf, x, y):
    mse_10 = cross_val_score(clf, x, y, cv=cv_10,
scoring='neg_mean_squared_error').mean()
    r2_10 = cross_val_score(clf, x, y, cv=cv_10, scoring='r2').mean()
    mse_5 = cross_val_score(clf, x, y, cv=cv_5,
scoring='neg_mean_squared_error').mean()
    r2_5 = cross_val_score(clf, x, y, cv=cv_5, scoring='r2').mean()
    return mse_10, r2_10, mse_5, r2_5

X = train_df2[['x1','x2']]
y = train_df2['v']

name_list = ['Linear Regression', 'Ridge Regression', 'Lasso Regression',
'ElasticNet',
         'Bayesian Ridge', 'SGD Regressor', 'Kernel Ridge',
         'K Neighbors Regressor', 'Decision Tree Regressor',
```

```
         'Random Forest Regressor', 'Gradient Boosting Regressor',
         'Adaboost Regressor', 'SVR', 'CatBoostRegressor',
         'XGB Regressor', 'LGBM Regressor']
clf_list = [LinearRegression(), Ridge(), Lasso(), ElasticNet(),
          BayesianRidge(), SGDRegressor(), KernelRidge(),
          KNeighborsRegressor(), DecisionTreeRegressor(),
          RandomForestRegressor(), GradientBoostingRegressor(),
          AdaBoostRegressor(), SVR(), CatBoostRegressor(),
          XGBRegressor(), LGBMRegressor()]

mse_10_list, r2_10_list, mse_5_list, r2_5_list = [], [], [], []
for clf in clf_list:
    # Get the best params for each clf

    mse_10, r2_10, mse_5, r2_5 = score_clf(clf, X, y)

    mse_10_list.append(mse_10)
    r2_10_list.append(r2_10)
    mse_5_list.append(mse_5)
    r2_5_list.append(r2_5)

clf_results = pd.DataFrame({'Classifier': name_list,
                            'MSE@10': mse_10_list,
                            'R2@10': r2_10_list,
                            'MSE@5': mse_5_list,
                            'R2@5': r2_5_list})
clf_results.sort_values(by="MSE@10", ascending=False)

# Take the top five best classifiers and perform grid search
clf_gs_list, clf_gs_params, clf_best_score, clf_var = [], [], [], []

## GridSearch with CatBoosting Regressor
params = {'depth':[2,3,5,7],
          'iterations':[250,100,500],
          'learning_rate':[0.001,0.01,0.05,0.1],
          'l2_leaf_reg':[1,3,5,10,100],
}
grid_cv = GridSearchCV(CatBoostRegressor(verbose=False), params, cv=cv_10,
scoring='neg_mean_squared_error', n_jobs=-1)
grid_cv.fit(X,y)
print(f"Best params: {grid_cv.best_params_}")
print(f"Best score: {grid_cv.best_score_}")
clf_gs_list.append('CatBoostRegressor')
clf_gs_params.append(grid_cv.best_params_)
clf_best_score.append(grid_cv.best_score_)
estimated_var = cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_10,
scoring='neg_mean_squared_error').var()
clf_var.append(estimated_var)
print(estimated_var)
print(cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_10,
scoring='neg_mean_squared_error').mean())
print(cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_5,
scoring='neg_mean_squared_error').mean())

## GridSearch with LGBM Regressor
params = {
    'num_leaves': [7, 14, 21, 28, 31, 50],
```

```
    'learning_rate':[0.001,0.01,0.05,0.1,0.2,0.3],
    'max_depth': [-1, 3, 5],
    'n_estimators': [50, 100, 200, 500],
}
grid_cv = GridSearchCV(LGBMRegressor(), params, cv=cv_10,
scoring='neg_mean_squared_error', n_jobs=-1)
grid_cv.fit(X,y)
print(f"Best params: {grid_cv.best_params_}")
print(f"Best score: {grid_cv.best_score_}")
clf_gs_list.append('LGBM Regressor')
clf_gs_params.append(grid_cv.best_params_)
clf_best_score.append(grid_cv.best_score_)
estimated_var = cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_10,
scoring='neg_mean_squared_error').var()
clf_var.append(estimated_var)
print(estimated_var)
print(cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_10,
scoring='neg_mean_squared_error').mean())
print(cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_5,
scoring='neg_mean_squared_error').mean())

## GridSearch with GradientBoosting Regressor
params = {
    'learning_rate': [0.001,0.01,0.1,1.0],
    'subsample'    : [0.9, 0.5, 0.2, 0.1],
    'n_estimators' : [10, 50, 100,500],
    'max_depth'    : [4,6,8]
}
grid_cv = GridSearchCV(GradientBoostingRegressor(), params, cv=cv_10,
scoring='neg_mean_squared_error', n_jobs=-1)
grid_cv.fit(X,y)
print(f"Best params: {grid_cv.best_params_}")
print(f"Best score: {grid_cv.best_score_}")
clf_gs_list.append('Gradient Boosting Regressor')
clf_gs_params.append(grid_cv.best_params_)
clf_best_score.append(grid_cv.best_score_)
estimated_var = cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_10,
scoring='neg_mean_squared_error').var()
clf_var.append(estimated_var)
print(estimated_var)
print(cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_10,
scoring='neg_mean_squared_error').mean())
print(cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_5,
scoring='neg_mean_squared_error').mean())

## GridSearch with KNeighbors
params = {
    'weights': ['uniform','distance'],
    'n_neighbors': [3,5,7,9,11,13,15,21]
}
grid_cv = GridSearchCV(KNeighborsRegressor(), params, cv=cv_10,
scoring='neg_mean_squared_error', n_jobs=-1)
grid_cv.fit(X,y)
print(f"Best params: {grid_cv.best_params_}")
print(f"Best score: {grid_cv.best_score_}")
clf_gs_list.append('KNeighborsRegressor')
clf_gs_params.append(grid_cv.best_params_)
```

```
clf_best_score.append(grid_cv.best_score_)
estimated_var = cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_10,
scoring='neg_mean_squared_error').var()
clf_var.append(estimated_var)
print(estimated_var)
print(cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_10,
scoring='neg_mean_squared_error').mean())
print(cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_5,
scoring='neg_mean_squared_error').mean())

## GridSearch with XGB Regressor
params = {
    "n_estimators": [50, 100, 300, 500],
    "learning_rate": (0.05, 0.10, 0.15),
    "max_depth": [ 3, 4, 5, 6, 7],
    "gamma":[ 0.0, 0.1, 0.2],
}
grid_cv = GridSearchCV(XGBRegressor(), params, cv=cv_10,
scoring='neg_mean_squared_error', n_jobs=-1)
grid_cv.fit(X,y)
print(f"Best params: {grid_cv.best_params_}")
print(f"Best score: {grid_cv.best_score_}")
clf_gs_list.append('XGBRegressor')
clf_gs_params.append(grid_cv.best_params_)
clf_best_score.append(grid_cv.best_score_)
estimated_var = cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_10,
scoring='neg_mean_squared_error').var()
clf_var.append(estimated_var)
print(estimated_var)
print(cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_10,
scoring='neg_mean_squared_error').mean())
print(cross_val_score(grid_cv.best_estimator_, X, y, cv=cv_5,
scoring='neg_mean_squared_error').mean())

clf_gs_results = pd.DataFrame({'Classifier': clf_gs_list,
                               'Best Params': clf_gs_params,
                               'Best Score@10': clf_best_score,
                               'Variance@10': clf_var})
clf_gs_results

testX1 = test_df.iloc[:,0]
testX2 = test_df.iloc[:,1]
test_df2 = pd.DataFrame({'x1':testX1, 'x2':testX2})
test_df2

# Best estimator for mu
best_clf_mu = SVR(C=1000, gamma='scale', kernel='rbf')
best_clf_mu.fit(X,y=train_df2['mu'])

# Predict on test data
mu_pred = best_clf_mu.predict(test_df2)

# Best estimator for var
best_clf_var = XGBRegressor(gamma=0.0, learning_rate=0.05, max_depth=3,
n_estimators=300)
best_clf_var.fit(X,y=train_df2['v'])
```

```
# Predict on test data
var_pred = best_clf_var.predict(test_df2)

test_df2['muhat'] = mu_pred
test_df2['Vhat'] = var_pred

# Save results
test_df2.to_csv("1.Cheng.Derek.csv",index=False,header=False)

# Ensure read properly
temp = pd.read_csv("1.Cheng.Derek.csv", header=None)
temp
```