Machine Learning

Regression - Assignment 3

By: Asaf Ben-Menachem
ID: 313352049
Asaf.Ben.Menachen@imprtva.com

Introduction

At first I thought of showing all my code with all the graphs and other plots.
My code has almost 500 lines and around 20 functions
It came out to be around 20 pages.
So I think it's better to explain what is my 'pipeline' and to explain each function what it does and what part it takes in order to create the best model and to be as modular as possible.

CORE

My script has the main() function but before this he divides the two different y's into 'Positive Tests' and 'Not Positive Tests' and does some basic data cleaning (the data was very good in the beginning) .

Main

The main function does the basic work that needed for all the regressions

- backward elimination to select the best features to predict the y
- How to calculate correlation between all columns
- Split the data to x_train, x_test, y_train, y_test witch will be used in all regressions
- StandardScaler for the X

And then the script go over all the regressions that we have and added to a data frame the indicators and will print it at the end of the script
I will show here my main function.

```
def main(X, cu_y):
    y = cu_y
    X = backward_elimination(X, y)
    X = remove_columns_corr(X, 0.95)
    #data_analysis_raw_data(X)
    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2)
    scaler = preprocessing.StandardScaler()
    x_train = scaler.fit_transform(x_train)
    x_test = scaler.transform(x_test)
    #  Linear Regression
    results_df = linear_reg(x_train, x_test, y_train, y_test)
    # Polynomial Regression.
    min_d = degree_chooser(x_train, x_test, y_train, y_test)
    results_df = poly_reg(x_train, x_test, y_train, y_test, results_df,min_d)
    # Ridge_Reg
    results_df = ridge_reg(x_train, x_test, y_train, y_test, results_df)
    # Lasso_Regression
    results_df = lasso_reg(x_train, x_test, y_train, y_test, results_df)
    # Random Forest Regressor.
```

```
    results_df = random_forrest(x_train, x_test, y_train,
y_test, results_df)
    # KNeighbors Regressor.
    results_df = kn_reg(x_train, x_test, y_train, y_test,
results_df)
    # Models Comparison
    results_df.set_index('Model', inplace=True)
    print(results_df)
```

To understand the function I will choose the polynomial regression as an example. (in general my regression functions are build in the same way.. But its very easy to understand small changes that I did)

Description one of the regressions

```
def poly_reg(x_train, x_test, y_train, y_test, results_df,
min_d) -> pd.DataFrame:
```

as we can see this function gets all their regular argument with special argument that holds what is the degrees that needed to be in set this version (min_d)
Min_d was created in the main function according to other function.
Most of the other regressions are using inner functions such as

```
def best_parameters_lasso(x_train, y_train) -> float:
    """this function use the GridSearchCV library tp find the
best alpha for the lasso regression"""
    model = Lasso(max_iter=11111)
    alpha = dict()
```

```
    alpha['alpha'] = np.arange(0.001, 1, 0.001)  # jump of
0.001 its the best for ridge

    search= GridSearchCV(estimator=model, param_grid=alpha,
scoring=score, cv=cv, n_jobs=-1)

    results = search.fit(x_train, y_train)

    return search.best_params_['alpha']
```

We will return to this polynomial regression function

```
def poly_reg(x_train, x_test, y_train, y_test, results_df, min_d) -> pd.DataFrame:
    """
    polynomail_reg predicted according to Regression models a target prediction value based on independent variables

    :rtype:
    :param results_df: has indicators for the other regression until now
    :param x_train: This includes your all independent variables,these will be used to train the model
    :param x_test: This is remaining  portion of the independent variables from the data,
           will be used to make predictions to test the accuracy of the model.
    :param y_train: This is your dependent variable which needs to be predicted by this model
    :param y_test: This data has category labels for your test data,
           these labels will be used to test the accuracy between actual and predicted categories.

    :return: pd.DataFrame that hold the indices of regression quality for the Regression
             in addition to the other regressions indices
    """
    poly_reg = PolynomialFeatures(degree=min_d)
    x_train_2_d = poly_reg.fit_transform(x_train)
    x_test_2_d = poly_reg.transform(x_test)
    lin_reg = LinearRegression()
    lin_reg.fit(x_train_2_d, y_train)

    test_pred = lin_reg.predict(x_test_2_d)

    results_df_2 = pd.DataFrame(data=[["Polynomail Regression", *evaluate(y_test, test_pred), cross_val(lin_reg)]],
                                columns=['Model', 'MAE', 'MSE', 'RMSE', 'R2 Square', 'Cross Validation'])
    results_df = results_df.append(results_df_2, ignore_index=True)
    #data_analysis_models_output(y_test,test_pred)
    #print_evaluate(y_test,test_pred)

    return results_df
```

The code is very straightforward and does exactly as was learned in class..
The only speasl thing that I added is a data frame that is called results_df_2 that creates all the indicators and appends it to the results_df that where all the indicators are.

Every function is build in with the functions

- data_analysis_models_output(y_test,test_pred) that create a plot to the regression + shows what is the y_test.

- print_evaluate(y_test,test_pred): that function prints the indicators of this reggrison.

  To use that the user needs to “turn them on” by commenting those lines.

Results

We can see the results of the regressions.
This on is for the Positive Tests

	MAE	MSE	RMSE	R2 Square	Cross Validation
Linear Regression	215.39101	84030.12879	289.87951	0.75584	0.709602677443389
Polynomail Regression	181.50275	51835.84801	227.67487	0.84938	0.709602677443389
Ridge Regression	216.38906	84625.26074	290.90421	0.75411	0.7096035373245437
Lasso Regression	347.80399	251680.17130	501.67736	0.26871	0.3107856993867083
Random Forest Regressor	136.24741	52567.34433	229.27570	0.84726	0.8592956044751597
KNeighbors Regressor	129.90052	41366.22513	203.38689	0.87981	NAN

And this one is for the Not Positive Tests

	MAE	MSE	RMSE	R2 Square	Cross Validation
Linear Regression	564.47239	513098.57996	716.30900	0.98897	0.9981064413930806
Polynomail Regression	486.36549	374177.81743	611.70076	0.99196	0.9981064413930806
Ridge Regression	561.74758	511063.06970	714.88675	0.98901	0.9981064450048187
Lasso Regression	563.87146	536386.33897	732.38401	0.98847	0.9957051604315105
andom Forest Regressor	397.26287	316550.79754	562.62847	0.99320	0.9972713204554436
KNeighbors Regressor	525.78010	548375.25654	740.52364	0.98821	NAN