Machine Learning (Regression) - Mini Project Report

Jeeva Mary Loui

Metro College Of Technology

Temperature Estimation of Permanent Magnent Synchronous motor

Abstract

To find the best model to estimate the temperature at various parts of permanat magnet synchronous motor different regression methods are run, which are Linear Regression , K-Nearest neeighbours , Random Forest , Adaboost and Support Vector Machine. Their performance are evaluated and the best model suited is found.

Machine Learning (Regression) - Mini Project Report

Dataset

Source: Kaggle.

Source url : https://www.kaggle.com/wkirgsn/electric-motor-temperature

The rotor temperature of the permanent magnet synchronous motor is monitored in real time by detecting variables, such as a three-phase current, a line voltage, a rotor position and speed and stator temperature, of the permanent magnet synchronous motor.

The dataset has 998070 rows and 13 columns.

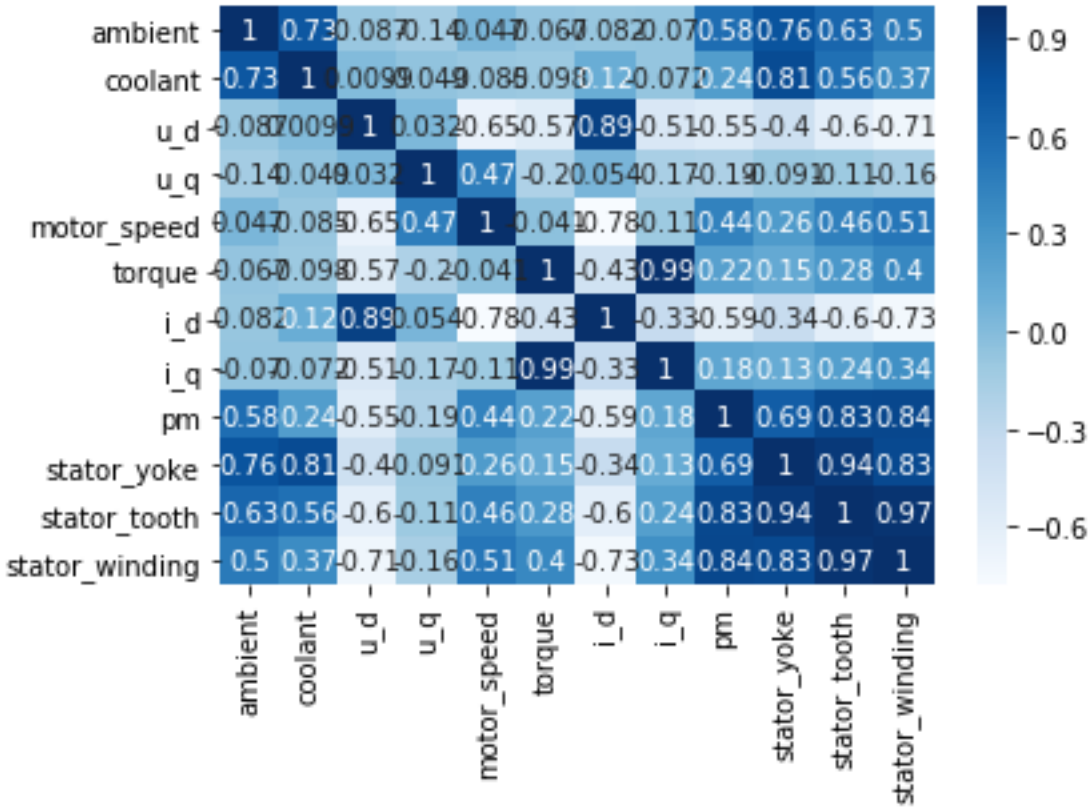
Data Preprocessing[1]

The size of datasetis so huge that it might cause very slow running of the regression techniques.Also the profile id shows each easurement session and are unique, mixing this causes error so profile_id which has maxium instance is chosen and all other rows are dropped.Then the size is again reduced by taking a sample of three percent of the total rows (pmsm = pmsm.sample(frac=0.10)) which reduced the size of dataset to (4397, 12).

The dataset is so clean that there are no null values .

Correlation Analysis.

For succesfulregression the features with strong correlation with target varible can be selcted using correlation analysis.Seaborn heatmap is an efficient method to study the correlation between variables.



The features having high correlations are represented with dark blue color as the color scale depicted in the heatmap.

The features ['pm', 'stator_tooth', 'stator_yoke', 'stator_winding'] are the targets in which the temperature estimation in stator winding is the most important feature. Thus the features and target is sepereated.

Recursive Feature Elimination.

Many Features can be elimated by using RFE which is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached.

The optimal number of features is found to be 7.

```
X_sub = X.loc[:,selected_features]
```

```
X_train, X_test, y_train, y_test = train_test_split(X_sub,y,random_state=0)
```

The features and target are split to test and train sets to run the model and evaluate it.

Linear Regression

From scikit learn library Linear Regression and the model is fit with the training set.

```
model = LinearRegression()
model.fit(X_train,y_train)
model.coef
```

The model is evaluated to get an accuracy score of 0.8136168044301262

```
model.score(X_test,y_test)
0.8136168044301262

model.score(X_train,y_train)
0.8128150194169108
```

The train and test scores are equal and no regulariztion is required.

Kfold Cross Validation

In K Fold cross validation, the data is divided into k subsets and train our model on k-1 subsets and hold the last one for test. This process is repeated k times, such that each time, one of the k subsets is used as the test set/ validation set and the other k-1 subsets are put together to form a training set. We then average the model against each of the folds and then finalize our model.

Doing K fold cross validation with linear regression accuracy of : 81.32454678394045

```
kf = KFold(n_splits=10, shuffle=True, random_state=1)
result=cross_val_score(model,X,y,cv=kf)
print(result)
print("Accuracy: "+ str(result.mean()*100))
```

Thus the accuracy is same with and without K-Fold Cross Validation.

K-Nearest NeighborsRegressor

Model is trained using KneighborsRegressor. And the R2 score is 0.965693.

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsRegressor()

knn.fit(X_train,y_train)
```

knn.score(X_test,y_test)
Out[90]: 0.9656928579716533
knn.score(X_train,y_train)
Out[91]: 0.9725267581330701

Train and test results are almost same and no signs of overfitting.

Grid Search Cross Validation:

To check the improvement of performance with cross validation Grid Search Cross Validation is used.

from sklearn.model_selection import GridSearchCV

For K-nearest neighbours the best parameters are found to be,

model.best_params_

2

The model's best score is **:** model.best_score_
Out[46]: 0.9536331178828906

To evaluate the scores and to do performance calculation two functions are defined:

```
def evaluation(model,X_test,y_test,pred):
    mse = mean_squared_error(y_test,pred)
    r2=model.score(X_test,y_test)
    print("R2 score "+str(r2))
    train_score=model.score(X_train,y_train)
    print("Train score "+str(train_score))
    print("Mean Squared Error "+str(mse))
#model fit function----------------------------------------------------------
def model_run(model):
    model.fit(X_test,y_test)
    pred=model.predict(X_test)
    evaluation(model,X_test,y_test,pred)
    return model.score(X_test,y_test)
```

Adaboost For Classification

Adaptive boosting is an ensemble method used for regression.

From sklearn.ensemble library Adaboost Forest Regressor is imported and the model is run.

```
ada=AdaBoostRegressor(n_estimators=500,learning_rate=0.001,random_state=1)
ada_sc=model_run(ada)
```

And the scores are

R2 score 0.8675429862168887
Train score 0.8419308672068194
Mean Squared Error 0.16246618248199415

Grid Search Cross Validation:

The adaboost regressor is run with gridserach cross validation with cross validation equal to 4.

adb.best_params_

{'learning_rate': 2, 'n_estimators': 750}

Mean training scores: 0.6262284473109639

Mean test scores :0.6138114825102591

The model best score is .26 which is fine and also the train score is only a little greater than test score score so there isno much problem of overfitting and regularization is not required.

Random Forest Classification

Random forest is an ensemble method used for classification and regression .

From sklearn.ensemble library Random Forest Regressor is imported and the model is run.

```
rfr=RandomForestRegressor(max_depth=4, random_state=2)
rfr_sc=model_run(rfr)
score_d.update( RandomForestRegressor = rfr_sc)
```

The R2 score is 0.9168418859714543

Grid Search CV

Running Grid Search CV for random forest

```
param_dict = {   'n_estimators':[10] ,
             'max_depth':[1,5,10,15,25]
```

```
                    }
rfr= RandomForestRegressor()
model = GridSearchCV(rfr,param_grid=param_dict,cv=4)
model.fit(X_train,y_train)
```

model.best_params_
{'max_depth': 25, 'n_estimators': 10}\
Mean training scores 0.8898652524379967
Mean test scores 0.8799319368821251

The model best score is . 0.9839558577471914 and also the train and test scores are equal . Hence Random forest could possibly be a good model.

Support Vector Machine for Classification

SVM is another method for Regression. From Scikit learn library SVM is imported and Support Vector Regressor is used to train a regression model.

```
from sklearn.svm import SVR
clf = SVC()
clf.fit(X_train,y_train)
```

The performance score is 0.807821721232231

GridSearch CV

Doing Grid Search Cross validation for SVM ,

model.best_params_
{'max_depth': 25, 'n_estimators': 10}
Mean training scores 0.8898652524379967
Mean test scores 0.8799319368821251

The model best score is . 0.9839558577471914 and the train score and test score are almost equal thus there are no signs of overfiting.

OLS Regression

From OLS regression Rsquared and Adjusted R Squared are 81.2% , Which is moderatly good score.P values are also very low and equals zero thus no feature needs to be elimintaed in the model.

OLS Regression Results

Dep. Variable:	stator_winding	R-squared:	0.813
Model:	OLS	Adj. R-squared:	0.812
Method:	Least Squares	F-statistic:	2040.
Date:	Sat, 13 Jul 2019	Prob (F-statistic):	0.00
Time:	23:26:20	Log-Likelihood:	-2364.4
No. Observations:	3297	AIC:	4745.
Df Residuals:	3289	BIC:	4794.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.5713	0.029	-19.970	0.000	-0.627	-0.515
ambient	0.2856	0.015	18.447	0.000	0.255	0.316
coolant	0.3804	0.014	26.683	0.000	0.352	0.408
u_q	-0.5129	0.029	-17.484	0.000	-0.570	-0.455
motor_speed	1.0970	0.070	15.662	0.000	0.960	1.234
torque	4.0191	0.408	9.855	0.000	3.219	4.819
i_d	0.4899	0.087	5.646	0.000	0.320	0.660
i_q	-2.9344	0.353	-8.309	0.000	-3.627	-2.242

CONCLUSION

The score dictionary give the best performance for K-Neighbours Regressor also no regulariztion is required

Key	Type	Size	Value
AdaBoostRegressor	float64	1	0.8675429862168887
KNeighborsRegressor	float64	1	0.9656928579716533
LinearRegression	float64	1	0.8136168044301262
RandomForestRegressor	float64	1	0.9168418859714543
SupportVectorRegressor	float64	1	0.807821721232231