19 May 2019
Group 4

CSE 4062 Term Project

Delivery #6 – Predictive Analytics

1- Feature Selection

ANOVA				
#	**Feature Name**	**Description**	**Type**	**Scores**
1	Product Number	Which product?	Nominal	2,4737
2	Brand Number	Which brand?	Nominal	17,3404
3	Profile Number	Which profile? (adult or kid)	Nominal	4,1

Table 1 – ANOVA measurements with categorical attributes and continuous target

MUTUAL INFORMATION REGRESSION				
#	**Feature Name**	**Description**	**Type**	**Mutual Information**
1	Product Number	Which product?	Nominal	12,011
2	Brand Number	Which brand?	Nominal	0,4848
3	Profile Number	Which profile? (adult or kid)	Nominal	0,065
4	Day	Day of sale	Nominal	0,00046
5	Month	Month of sale	Nominal	0,0073
6	Year	Year of sale	Nominal	0,0161
7	Total Sales Revenue (₺)	Total revenue of the product on given date	Numeric	2,122
8	Unit Price	Unit price of the product	Numeric	0,8412
9	Change in Inflation (monthly)	Change in monthly inflation rate in Turkey	Numeric	0,036

Table 2 – Mutual Information for Regression

2- Regression Experiments

LINEAR REGRESSION				
#	**Experiment**	**Mean Squared Error**	**Mean Absolute Error**	**R-Square Score**
1	train_test_split with test_size = 0.2	106980.95	201.48	0.69
2	train_test_split with test_size = 0.3	110907.25	203.04	0.69
3	train_test_split with test_size = 0.4	111147.67	202.80	0.69
4	train_test_split with test_size = 0.5	110444.71	202.64	0.69
5	k_fold with k = 5	98067.48	200.65	0.72
6	k_fold with k = 6	94775.70	200.73	0.74
7	k_fold with k = 7	94797.43	199.80	0.73
8	k_fold with k = 8	91375.94	200.32	0.74
9	k_fold with k = 9	91951.43	201.23	0.75
10	k_fold with k = 10	98892.48	201.82	0.75
** With K-Fold method, best resulting model selected accross k splits on data.*				
** Random state is given using numpy.rand.randint(500) and its same for all experiments.*				

Table 3 – Linear Regression experiments

POLYNOMIAL REGRESSION *(POLYNOMIAL DEGREE = 2)*				
#	**Experiment**	**Mean Squared Error**	**Mean Absolute Error**	**R-Square Score**
1	train_test_split with test_size = 0.2	92082.78	186.28	0.76
2	train_test_split with test_size = 0.3	93372.40	185.32	0.75
3	train_test_split with test_size = 0.4	90867.51	183.70	0.75
4	train_test_split with test_size = 0.5	87811.96	183.46	0.76
5	k_fold with k = 5	75194.10	179.22	0.78
6	k_fold with k = 6	75929.42	180.17	0.78
7	k_fold with k = 7	73909.80	182.47	0.79
8	k_fold with k = 8	71428.47	182.32	0.80
9	k_fold with k = 9	70781.06	178.59	0.79
10	k_fold with k = 10	71008.16	179.13	0.78
** With K-Fold method, best resulting model selected accross k splits on data.*				
** Random state is given using numpy.rand.randint(500) and its same for all experiments.*				

Table 4 – Polynomial Regression experiments with polynomial degree = 2

POLYNOMIAL REGRESSION *(POLYNOMIAL DEGREE = 3)*				
#	**Experiment**	**Mean Squared Error**	**Mean Absolute Error**	**R-Square Score**
1	train_test_split with test_size = 0.2	67135.23	155.39	0.80
2	train_test_split with test_size = 0.3	69384.95	155.37	0.80
3	train_test_split with test_size = 0.4	72239.87	156.17	0.79
4	train_test_split with test_size = 0.5	71073.34	155.93	0.80
5	k_fold with k = 5	61326.38	152.75	0.81
6	k_fold with k = 6	61227.58	151.70	0.81
7	k_fold with k = 7	62131.98	154.56	0.81
8	k_fold with k = 8	61056.43	154.4	0.82
9	k_fold with k = 9	60074.49	154.47	0.82
10	k_fold with k = 10	61097.59	149.14	0.81
** With K-Fold method, best resulting model selected accross k splits on data.*				
** Random state is given using numpy.rand.randint(500) and its same for all experiments.*				

Table 5 – Polynomial Regression experiments with polynomial degree = 3

POLYNOMIAL REGRESSION *(POLYNOMIAL DEGREE = 4)*				
#	**Experiment**	**Mean Squared Error**	**Mean Absolute Error**	**R-Square Score**
1	train_test_split with test_size = 0.2	67469.20	150.69	0.81
2	train_test_split with test_size = 0.3	65523.28	150.49	0.81
3	train_test_split with test_size = 0.4	67200.50	150.31	0.81
4	train_test_split with test_size = 0.5	67355.01	150.13	0.81
5	k_fold with k = 5	62203.70	150.40	0.81
6	k_fold with k = 6	59671.49	146.08	0.83
7	k_fold with k = 7	60943.03	149.85	0.83
8	k_fold with k = 8	61908.80	148.05	0.82
9	k_fold with k = 9	55186.84	145.55	0.83
10	k_fold with k = 10	56228.49	144.93	0.83
** With K-Fold method, best resulting model selected accross k splits on data.*				
** Random state is given using numpy.rand.randint(500) and its same for all experiments.*				

Table 6 – Polynomial Regression experiments with polynomial degree = 4

POLYNOMIAL REGRESSION *(POLYNOMIAL DEGREE = 5)*				
#	**Experiment**	**Mean Squared Error**	**Mean Absolute Error**	**R-Square Score**
1	train_test_split with test_size = 0.2	59304.17	148.30	0.84
2	train_test_split with test_size = 0.3	64235.32	147.81	0.82
3	train_test_split with test_size = 0.4	64639.50	148.03	0.82
4	train_test_split with test_size = 0.5	63756.77	148.19	0.82
5	k_fold with k = 5	60873.59	148.06	0.82
6	k_fold with k = 6	59656.65	148.42	0.82
7	k_fold with k = 7	61693.03	145.82	0.82
8	k_fold with k = 8	58788.70	146.83	0.84
9	k_fold with k = 9	60150.97	146.99	0.83
10	k_fold with k = 10	56071.21	147.59	0.82
** With K-Fold method, best resulting model selected accross k splits on data.*				
** Random state is given using numpy.rand.randint(500) and its same for all experiments.*				

Table 7 – Polynomial Regression experiments with polynomial degree = 5

STOCHASTIC GRADIENT DESCENT				
#	**Experiment**	**Mean Squared Error**	**Mean Absolute Error**	**R-Square Score**

1	train_test_split with test_size=0.3
	alpha=1e-4
	max_iter=1000
		2,78E+46	1088728319845761.9	-7,63E+38

2	train_test_split with test_size=0.3
	alpha=1e-3
	max_iter=1000
		1,35E+46	915178551426498.8	-3,70E+40

3	train_test_split with test_size=0.3
	alpha=1e-2
	max_iter=1000
		5,67E+47	1,56E+32	-1,56E+42

4	train_test_split with test_size=0.3
	alpha=1e-1
	max_iter=1000
		3,52E+46	1396274709447520.8	-9,66E+39

5	train_test_split with test_size=0.3
	alpha=1
	max_iter=1000
		2,99E+46	1,14045E+15	-8,22E+39

** Random state is given using numpy.rand.randint(500) and its same for all experiments.*				
** Out data didn't fit into stochastic gradient descent. Algorithm couldn't converge to optimal solution with given iterations.*				

Table 8 – Stochastic Gradient Descent Regressor experiments

LINEAR SVR				
#	**Experiment**	**Mean Squared Error**	**Mean Absolute Error**	**R-Square Score**

1	train_test_split with
	test_size=0.3
	C=1e-2
	max_iter=1000
		105378.68	197.39	0.69

2	train_test_split with
	test_size=0.3
	C=1e-1
	max_iter=1000
		192546.62	343.68	0.44

3	train_test_split with
	test_size=0.3
	C=1
	max_iter=1000
		121095.43	201.49	0.65

4	train_test_split with
	test_size=0.3
	C=10
	max_iter=1000
		121095.43	201.49	0.65

5	train_test_split with
	test_size=0.3
	C=100
	max_iter=1000
		121095.43	201.49	0.65

** Random state is given using numpy.rand.randint(500) and its same for all experiments.*				
** Out data didn't fit into linear SVR. Algorithm couldn't converge to optimal solution with given iterations.*				

Table 9 – Linear Support Vector Regressor experiments

3- Results

For this phase, we are required to create a predictive model for the data. Since our target is continuous, we have tested regression models and choose the best model among all models that we have tested.

We used all features for training step, which are Product Number, Brand Number, Profile Number, Date of sale, Total Sales Revenue, Unit Price, Monthly Inflation Changing Rate in Turkey and we are trying to predict total sales amount of a product with given date and approximative change in inflation rate.

We tested 4 types of regression algorithms while evaluating models which are linear regression, polynomial regression, linear support vector regressor and stochastic gradient descent regressor.

- We tested linear regression algorithm with different cross validation techniques which are train-test split of scikit learn (splits data as train data and test data with given train or test sizes) and k fold (splits data to k equal pieces; leaves one piece out as test set and remaining as training set, iterates k times and tries all pieces). We get mean squared errors, mean absolute errors and R square scores of all models and choose best model according to these metrics.
- We tested polynomial regression algorithms with same cross validation techniques used in linear regression. Polynomial regression needs a parameter 'degree' which defines polynomial degree of regressor. In our case, we choose polynomial features as change in inflation rate and unit price of product. Among degrees 2, 3, 4 and 5, we observed that higher degree gives better performance of prediction, but it increases complexity and variance of the model.

- We tested linear SVR algorithm with different penalty (C) parameters which effect the regularization amount (overfit avoidance technique). As we know, SVR algorithm isn't a good choice for data having more than 10.000 instances [1]. Unfortunately, our observations show us that our data doesn't fit into linear SVR model well and algorithm couldn't converge to optimal solution with given iterations. So we eliminated linear SVR model for our problem.
- We tested SGD(Stochastic Gradient Descent) algorithm with different alpha parameters which also effect the regularization term. As we can see from Table 8, our data couldn't fit into SGD model. Algorithm couldn't converge the optimum solution with given iterations as we can understand from unrealistic metric results.

While evaluating models, we had 3 metrics. Mean squared error and mean absolute error means better predictive performance when they are small. Their values have no upper or lower bound. But R square score means worse predictive performance when it's small, and also its a value between 0 and 1. We combined these 3 metrics to obtain best model between all algorithms with different parameters by this way: We normalized all mean squared errors and mean absolute errors so that they can be between 0 and 1. After that, we subtracted calculated R square scores from 1, then we sum all scores, then we took the minimum of these values. Index of the minimum value also shows us the best fitting model to data. In our case, we obtained the best model as a polynomial regression model with polynomial degree of 5.

For prediction phase, best model expects number of the product, day, month and year of sale in a future time, estimated monthly inflation rate change. While training our model, we used unit price and total sales revenue columns and it's not possible for the user to give these informations for prediction. Because of that, we also tried to predict unit price/total sales revenue of given product number at given day with years(2016, 2017, 2018 maximum) and previous change in inflation rate with a linear regression model. Since we have at max 3 instances(sales of given product at given day, given month and years 2016, 2017, 2018) and 2 features (year and change in inflation rate), it's not so possible to predict total sales revenue and unit price. So, we decided to get mean of previous total sales revenue and unit price of given product at given day, month and years 2016, 2017, 2018 as current total sales revenue/unit price. After that, we have all attributes needed for prediction (product number, brand number, profile number, date of sale, total sales revenue, unit price and change in inflation rate) of total sales amount.

If the best model chosen as a polynomial regression model, in our case it is, we are required to extend features unit price and inflation rate with polynomial degree of the chosen model. After that, model is ready to make predictions with given parameters 'Product Number', 'Day, Month and Year' and 'Estimated Change in Monthly Inflation Rate'. After tests, we had meaningful results – e.g. increase in inflation caused less sales amount.

** Model predicted some negative sales amounts for some parameters, which is not possible, we can conclude it would be better if we don't sell given product anymore according to model.*

4- References

[1] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html