Exercise: Customer Spending Prediction

Use numeric prediction techniques to build a predictive model for the HW3.xlsx dataset. This dataset is provided on the course website and contains data about whether or not different consumers made a purchase in response to a test mailing of a certain catalog and, in case of a purchase, how much money each consumer spent. The data file has a brief description of all the attributes in a separate worksheet. Note that this dataset has two possible outcome variables: Purchase (0/1 value: whether or not the purchase was made) and Spending (numeric value: amount spent).

Use Rapidminer for this Exercise.

- **(a) Build numeric prediction models that predict Spending based on the other available customer information (obviously, not including the Purchase attribute among the inputs!). Use linear regression, k-NN, and regression tree techniques. Briefly discuss your explorations and present the best result (best predictive model) for each of the three techniques.**
- **(b) Compare the three techniques (i.e., your three best models); which of them provides the best predictive performance?**
- **(c) Also, discuss/interpret the linear regression model. Please make sure you use best practices for predictive modeling. (i.e., do you need to split data? etc.)**

(a)

1. Data exploration:

Name	Type	Missing	Statistics		
Label **Spending**	Real	0	Min 0	Max 1500.060	Average 102.561
US	Binominal	0	Least 0.0 (351)	Most 1.0 (1649)	Values 1.0 (1649), 0.0 (351)
source_a	Binominal	0	Least 1 (253)	Most 0 (1747)	Values 0 (1747), 1 (253)
source_c	Binominal	0	Least 1 (112)	Most 0 (1888)	Values 0 (1888), 1 (112)
source_b	Binominal	0	Least 1 (120)	Most 0 (1880)	Values 0 (1880), 1 (120)
source_d	Binominal	0	Least 1 (83)	Most 0 (1917)	Values 0 (1917), 1 (83)
source_e	Binominal	0	Least 1 (302)	Most 0 (1698)	Values 0 (1698), 1 (302)
source_m	Binominal	0	Least 1 (33)	Most 0 (1967)	Values 0 (1967), 1 (33)
source_o	Binominal	0	Least 1 (67)	Most 0 (1933)	Values 0 (1933), 1 (67)
source_h	Binominal	0	Least 1 (105)	Most 0 (1895)	Values 0 (1895), 1 (105)
source_r	Binominal	0	Least 1 (137)	Most 0 (1863)	Values 0 (1863), 1 (137)
source_s	Binominal	0	Least 1 (94)	Most 0 (1906)	Values 0 (1906), 1 (94)
source_t	Binominal	0	Least 1 (43)	Most 0 (1957)	Values 0 (1957), 1 (43)
source_u	Binominal	0	Least 1 (238)	Most 0 (1762)	Values 0 (1762), 1 (238)
source_p	Binominal	0	Least 1 (12)	Most 0 (1988)	Values 0 (1988), 1 (12)
source_x	Binominal	0	Least 1 (36)	Most 0 (1964)	Values 0 (1964), 1 (36)
source_w	Binominal	0	Least 1 (275)	Most 0 (1725)	Values 0 (1725), 1 (275)
Freq	Integer	0	Min 0	Max 15	Average 1.417
last_update_days_ago	Integer	0	Min 1	Max 4188	Average 2155.101
1st_update_days_ago	Integer	0	Min 1	Max 4188	Average 2435.602
Web order	Binominal	0	Least 1 (852)	Most 0 (1148)	Values 0 (1148), 1 (852)
Gender=male	Binominal	0	Least 0 (951)	Most 1 (1049)	Values 1 (1049), 0 (951)
Address_is_res	Binominal	0	Least 1 (442)	Most 0 (1558)	Values 0 (1558), 1 (442)

Overall, there are 19 binomial attributes and 4 numeric attributes (after excluding *sequence_number* and *Purchase*, and 2000 observations. We filtered out all missing value.

I use **10 fold cross validation** and **normalized** all attributes in RapidMiner to access these three models' performance.

2. Linear regression

2.1 use “T-Test” as feature selection criteria with alpha=5%. First, we need to turn nominal to numerical attributes.

```
PerformanceVector:
root_mean_squared_error: 127.625 +/- 21.000 (mikro: 129.341 +/- 0.000)
absolute_error: 78.023 +/- 5.989 (mikro: 78.023 +/- 103.158)
relative_error: 30,215.08% +/- 4,147.60% (mikro: 30,185.13% +/- 59,090.58%)
normalized_absolute_error: 0.681 +/- 0.079 (mikro: 0.673)
root_relative_squared_error: 0.706 +/- 0.096 (mikro: 0.695)
squared_error: 16729.118 +/- 5851.833 (mikro: 16729.118 +/- 90042.542)
```

Attribute	Coefficient	Std. Error	Std. Coefficient	Tolerance	t-Stat	p-Value	Code
source_c = 0	30.199	18.133	0.037	0.997	1.665	0.096	*
source_c = 1	-30.198	18.133	-0.037	0.997	-1.665	0.096	*
source_h = 0	57.899	18.855	0.069	0.980	3.071	0.002	***
source_h = 1	-57.900	18.855	-0.069	0.980	-3.071	0.002	***
Address_is_res = 1	-32.212	10.041	-0.072	0.999	-3.208	0.001	***
Address_is_res = 0	32.212	10.041	0.072	0.999	3.208	0.001	***
Freq	133.441	4.309	0.715	0.934	30.969	0	****
last_update_days_ago	-12.850	4.331	-0.069	0.924	-2.967	0.003	***
(Intercept)	5.950	∞	?	?	0	1	

2.2 use “M5 prime” as feature selection with alpha=5%

```
PerformanceVector:
root_mean_squared_error: 125.864 +/- 20.980 (mikro: 127.601 +/- 0.000)
absolute_error: 76.780 +/- 5.949 (mikro: 76.780 +/- 101.916)
relative_error: 28,499.43% +/- 3,063.02% (mikro: 28,474.56% +/- 58,129.74%)
normalized_absolute_error: 0.669 +/- 0.076 (mikro: 0.662)
root_relative_squared_error: 0.696 +/- 0.095 (mikro: 0.685)
squared_error: 16281.959 +/- 5795.089 (mikro: 16281.959 +/- 88052.510)
```

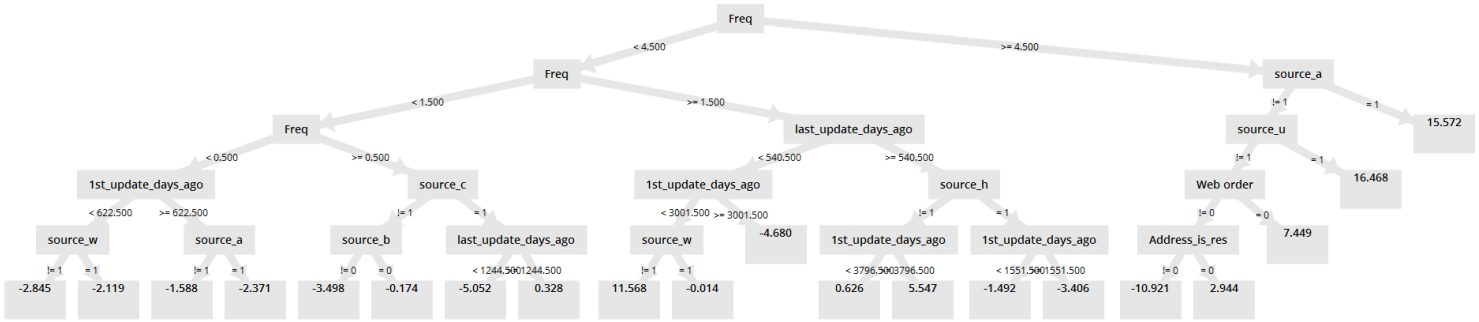
Attribute	Coefficient	Std. Error	Std. Coefficient	Tolerance	t-Stat	p-Value	Code
US = 1.0	-0.968	4.144	-0.005	1.000	-0.234	0.815	
US = 0.0	0.968	4.144	0.005	1.000	0.234	0.815	
source_a = 0	-4.934	4.279	-0.026	0.938	-1.153	0.249	
source_a = 1	4.931	4.279	0.026	0.938	1.152	0.249	
source_c = 0	6.049	4.149	0.032	0.997	1.458	0.145	
source_c = 1	-6.057	4.149	-0.032	0.997	-1.460	0.145	
source_b = 1	-0.854	4.157	-0.005	0.994	-0.205	0.837	
source_b = 0	0.859	4.157	0.005	0.994	0.207	0.836	
source_d = 0	2.500	4.144	0.013	1.000	0.603	0.546	
source_d = 1	-2.499	4.144	-0.013	1.000	-0.603	0.547	
source_e = 0	1.392	4.152	0.007	0.996	0.335	0.737	
source_e = 1	-1.388	4.152	-0.007	0.996	-0.334	0.738	
source_m = 0	0.985	4.144	0.005	1.000	0.238	0.812	
source_m = 1	-0.990	4.144	-0.005	1.000	-0.239	0.811	
source_o = 0	-1.694	4.162	-0.009	0.991	-0.407	0.684	
source_o = 1	1.693	4.162	0.009	0.991	0.407	0.684	
source_h = 0	11.593	4.185	0.062	0.981	2.770	0.006	***
source_h = 1	-11.593	4.185	-0.062	0.981	-2.771	0.006	***
source_r = 0	-4.438	4.157	-0.024	0.994	-1.068	0.286	
source_r = 1	4.439	4.157	0.024	0.994	1.068	0.286	
source_s = 0	1.284	4.159	0.007	0.993	0.309	0.758	
source_s = 1	-1.283	4.159	-0.007	0.993	-0.308	0.758	
source_t = 0	1.003	4.144	0.005	1.000	0.242	0.809	
source_t = 1	-1.005	4.144	-0.005	1.000	-0.242	0.808	
source_u = 0	-5.002	4.181	-0.027	0.982	-1.196	0.232	
source_u = 1	5.003	4.181	0.027	0.982	1.197	0.232	
source_p = 0	0.416	4.145	0.002	0.999	0.100	0.920	
source_p = 1	-0.415	4.145	-0.002	0.999	-0.100	0.920	
source_x = 0	-0.176	4.146	-0.001	0.999	-0.042	0.966	
Web order = 0	-5.750	4.177	-0.031	0.984	-1.377	0.169	
Gender=male = 0	0.754	4.146	0.004	0.999	0.182	0.856	
Gender=male = 1	-0.750	4.146	-0.004	0.999	-0.181	0.856	
Address_is_res = 1	-13.594	4.146	-0.073	0.999	-3.279	0.001	***
Address_is_res = 0	13.594	4.146	0.073	0.999	3.278	0.001	***
Freq	129.224	5.416	0.692	0.585	23.858	0	****
last_update_days_ago	-16.254	4.386	-0.087	0.893	-3.706	0.000	****
1st_update_days_ago	2.953	4.154	0.016	0.995	0.711	0.477	
(Intercept)	102.561	∞	?	?	0	1	

3. Decision Tree

3.1 Gradient Boosted Tree (max depth=5)

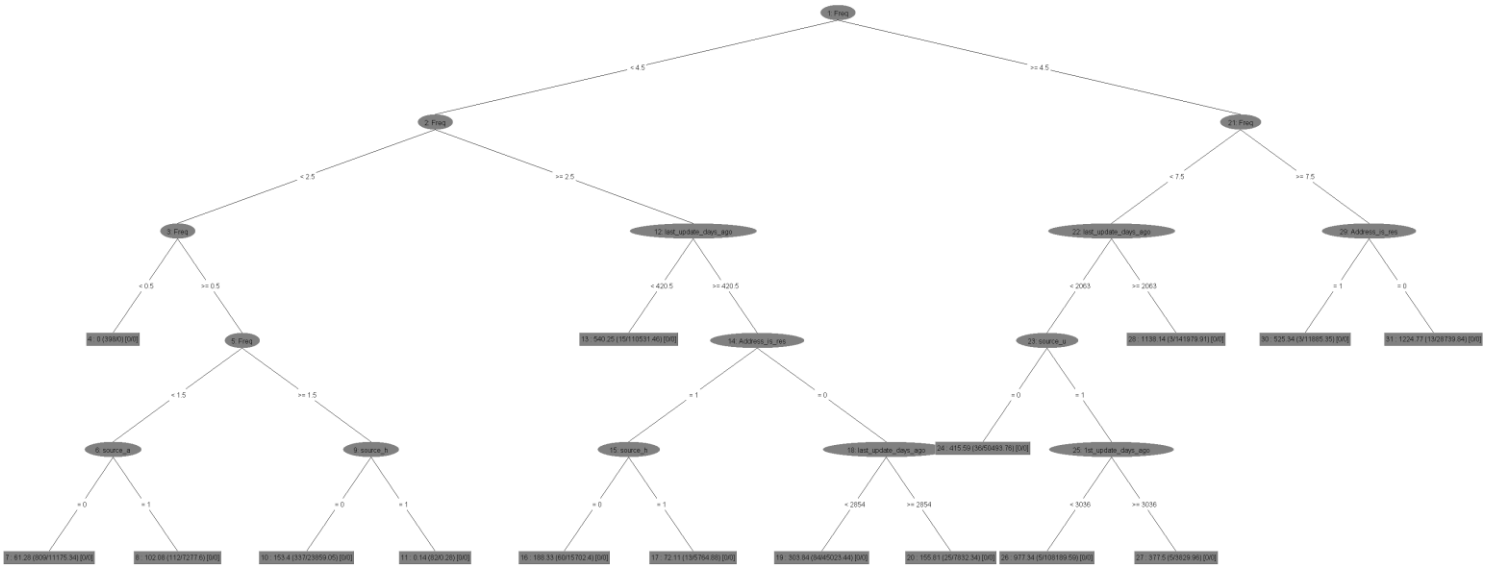
```
PerformanceVector:
root_mean_squared_error: 123.236 +/- 21.708 (mikro: 125.134 +/- 0.000)
absolute_error: 72.264 +/- 5.851 (mikro: 72.264 +/- 102.158)
relative_error: 30,302.63% +/- 3,768.24% (mikro: 30,271.53% +/- 54,707.08%)
normalized_absolute_error: 0.629 +/- 0.065 (mikro: 0.623)
root_relative_squared_error: 0.680 +/- 0.091 (mikro: 0.672)
squared_error: 15658.404 +/- 5848.392 (mikro: 15658.404 +/- 89750.402)
```

Tree 20:



3.2 Weka-REPTree (max depth=5).

```
PerformanceVector:
root_mean_squared_error: 134.358 +/- 24.238 (mikro: 136.527 +/- 0.000)
absolute_error: 72.973 +/- 7.950 (mikro: 72.973 +/- 115.389)
relative_error: 27,481.68% +/- 5,697.98% (mikro: 27,451.48% +/- 63,012.52%)
normalized_absolute_error: 0.633 +/- 0.054 (mikro: 0.629)
root_relative_squared_error: 0.742 +/- 0.101 (mikro: 0.733)
squared_error: 18639.612 +/- 7248.235 (mikro: 18639.612 +/- 96579.974)
```



W-REPTree

```
REPTree
============

Freq < 4.5
|   Freq < 2.5
|   |   Freq < 0.5 : 0 (398/0) [0/0]
|   |   Freq >= 0.5
|   |   |   Freq < 1.5
|   |   |   |   source_a = 0 : 61.28 (809/11175.34) [0/0]
|   |   |   |   source_a = 1 : 102.08 (112/7277.6) [0/0]
|   |   |   Freq >= 1.5
|   |   |   |   source_h = 0 : 153.4 (337/23859.05) [0/0]
|   |   |   |   source_h = 1 : 0.14 (82/0.28) [0/0]
|   Freq >= 2.5
|   |   last_update_days_ago < 420.5 : 540.25 (15/110531.46) [0/0]
|   |   last_update_days_ago >= 420.5
|   |   |   Address_is_res = 1
|   |   |   |   source_h = 0 : 188.33 (60/15702.4) [0/0]
|   |   |   |   source_h = 1 : 72.11 (13/5764.88) [0/0]
|   |   |   Address_is_res = 0
|   |   |   |   last_update_days_ago < 2854 : 303.84 (84/45023.44) [0/0]
|   |   |   |   last_update_days_ago >= 2854 : 155.81 (25/7832.34) [0/0]
Freq >= 4.5
|   Freq < 7.5
|   |   last_update_days_ago < 2063
|   |   |   source_u = 0 : 415.59 (36/50493.76) [0/0]
|   |   |   source_u = 1
|   |   |   |   lst_update_days_ago < 3036 : 977.34 (5/108189.59) [0/0]
|   |   |   |   lst_update_days_ago >= 3036 : 377.5 (5/3829.96) [0/0]
|   |   last_update_days_ago >= 2063 : 1138.14 (3/141979.91) [0/0]
|   Freq >= 7.5
|   |   Address_is_res = 1 : 525.34 (3/11885.35) [0/0]
|   |   Address_is_res = 0 : 1224.77 (13/28739.84) [0/0]

Size of the tree : 31
```

4. KNN

I normalize attributes to produce KNN model. After comparing performance vectors, we can conclude that k=10 is better than k=7.

4.1 KNN with normalization

K=7

```
PerformanceVector:
root_mean_squared_error: 132.595 +/- 22.600 (mikro: 134.507 +/- 0.000)
absolute_error: 71.019 +/- 6.761 (mikro: 71.019 +/- 114.230)
relative_error: 19,725.14% +/- 2,774.81% (mikro: 19,723.20% +/- 54,579.22%)
normalized_absolute_error: 0.617 +/- 0.060 (mikro: 0.613)
root_relative_squared_error: 0.734 +/- 0.109 (mikro: 0.722)
squared_error: 18092.082 +/- 6509.404 (mikro: 18092.082 +/- 103406.039)
```

K=10

```
PerformanceVector:
root_mean_squared_error: 129.616 +/- 23.190 (mikro: 131.674 +/- 0.000)
absolute_error: 70.013 +/- 6.764 (mikro: 70.013 +/- 111.518)
relative_error: 20,122.42% +/- 2,551.44% (mikro: 20,097.17% +/- 52,271.41%)
normalized_absolute_error: 0.608 +/- 0.056 (mikro: 0.604)
root_relative_squared_error: 0.715 +/- 0.092 (mikro: 0.707)
squared_error: 17337.972 +/- 6543.877 (mikro: 17337.972 +/- 101720.327)
```

4.2 KNN attribute selection-Forward selection

K=7

```
PerformanceVector:
root_mean_squared_error: 124.666 +/- 27.749 (mikro: 127.717 +/- 0.000)
absolute_error: 66.917 +/- 8.746 (mikro: 66.917 +/- 108.783)
relative_error: 21,817.49% +/- 6,353.75% (mikro: 21,839.69% +/- 57,339.91%)
normalized_absolute_error: 0.581 +/- 0.075 (mikro: 0.578)
squared_error: 16311.540 +/- 7063.060 (mikro: 16311.540 +/- 91041.131)
```

K=10

It uses forward selection to choose attributes and doesn't include *source_r* in the model.

```
PerformanceVector:
root_mean_squared_error: 123.307 +/- 34.556 (mikro: 128.057 +/- 0.000)
absolute_error: 67.566 +/- 10.689 (mikro: 67.566 +/- 108.782)
relative_error: 21,714.49% +/- 6,324.43% (mikro: 21,611.88% +/- 53,174.38%)
normalized_absolute_error: 0.585 +/- 0.040 (mikro: 0.584)
squared_error: 16398.664 +/- 9003.740 (mikro: 16398.664 +/- 92683.953)
```

attribute	weight
Freq	1
last_upd...	1
1st_upd...	0
US	0
source_a	1
source_c	1
source_b	0
source_d	0
source_e	0
source_m	1
source_o	0
source_h	1
source_r	1
source_s	0
source_t	0
source_u	1
source_p	0
source_x	0
source_w	0
Web order	0
Gender=...	0
Address...	1

4.3 KNN attribute selection-Backward selection

K=7

```
PerformanceVector:
root_mean_squared_error: 127.794 +/- 28.681 (mikro: 130.973 +/- 0.000)
absolute_error: 70.889 +/- 8.349 (mikro: 70.889 +/- 110.130)
relative_error: 20,897.37% +/- 5,226.87% (mikro: 20,875.58% +/- 57,003.74%)
normalized_absolute_error: 0.616 +/- 0.054 (mikro: 0.612)
squared_error: 17153.865 +/- 7861.418 (mikro: 17153.865 +/- 96033.429)
```

K=10

```
PerformanceVector:
root_mean_squared_error: 126.353 +/- 29.345 (mikro: 129.716 +/- 0.000)
absolute_error: 70.538 +/- 8.370 (mikro: 70.538 +/- 108.861)
relative_error: 20,865.49% +/- 4,587.96% (mikro: 20,850.44% +/- 53,477.00%)
normalized_absolute_error: 0.612 +/- 0.044 (mikro: 0.609)
squared_error: 16826.300 +/- 7808.809 (mikro: 16826.300 +/- 95881.092)
```

attribute	weight
Freq	1
last_upd...	1
1st_upd...	1
US	1
source_a	1
source_c	1
source_b	1
source_d	1
source_e	1
source_m	1
source_o	1
source_h	1
source_r	0
source_s	1
source_t	1
source_u	1
source_p	1
source_x	1
source_w	1
Web order	1
Gender=...	1
Address...	1

(b) compare models

Linear regression with feature selection and KNN with fwd or bwd selection selects the most important attributes into models.

We use root mean squared error and range from cross validation to choose the best model.

Apparently, KNN attribute selection-Forward selection (k=10) with root mean squared error=123.307 +/-34.556 is the best one.

(c)Linear regression

Normalize data and use 10-fold cross validation to access model's performance.

For example, we choose "T-Test" as feature selection criteria with alpha=5%. The following is result:

```
PerformanceVector:
root_mean_squared_error: 127.625 +/- 21.000 (mikro: 129.341 +/- 0.000)
absolute_error: 78.023 +/- 5.989 (mikro: 78.023 +/- 103.158)
relative_error: 30,215.08% +/- 4,147.60% (mikro: 30,185.13% +/- 59,090.58%)
normalized_absolute_error: 0.681 +/- 0.079 (mikro: 0.673)
root_relative_squared_error: 0.706 +/- 0.096 (mikro: 0.695)
squared_error: 16729.118 +/- 5851.833 (mikro: 16729.118 +/- 90042.542)
```

Attribute	Coefficient	Std. Error	Std. Coefficient	Tolerance	t-Stat	p-Value	Code
source_c = 0	30.199	18.133	0.037	0.997	1.665	0.096	*
source_c = 1	-30.198	18.133	-0.037	0.997	-1.665	0.096	*
source_h = 0	57.899	18.855	0.069	0.980	3.071	0.002	***
source_h = 1	-57.900	18.855	-0.069	0.980	-3.071	0.002	***
Address_is_res = 1	-32.212	10.041	-0.072	0.999	-3.208	0.001	***
Address_is_res = 0	32.212	10.041	0.072	0.999	3.208	0.001	***
Freq	133.441	4.309	0.715	0.934	30.969	0	****
last_update_days_ago	-12.850	4.331	-0.069	0.924	-2.967	0.003	***
(Intercept)	5.950	∞	?	?	0	1	

There are 8 predictors have significant influence on target value, of which *Source catalog C and H, Address is a residence,* and *# days ago was last update to cust. record* have negative relationship and others have positive relationship.
Freq (Number of transactions in last year at source catalo) has the largest weight in this model.
Spending = 30.199*no source_c – 30.198*yes source_c + 57.900*no source_h – 57.900* yes source_h -32.212* Add is res + 32.212* Add is not res +133.441* Freq -12.850* last_update_days_ago + 5.950