DATA MINING IN ACTION

Assignment 3

Kunanon Pititheerachot
12634123

Contents

The data mining problem, inputs, output

Firstly, for data mining problem, in “TrainingSet” csv file has several datasets has no value such as “add_these_pw_job_title_9089”, “application_type”, “case_no” and a lot more, which is not useful for my model and not possible to do date pre-processing. So, I considered these columns datasets as unmeaningful dataset and delete all columns that has no value. Secondly, after roughly exploring the rest datasets I can see some of datasets has a lot of missing data which will cause an error when fitting model, I also delete some of the datasets such as “foreign_worker_info_training_comp”, “foreign_worker_ownership_interest”, “naics_code”, and more that has amount of missing value similar to these datasets. Thirdly, I considered some datasets that not provide enough meaningful even if apply data preprocessing method such as “agent_city”, “agent_firm_name”, “agent_state”, and any similar datasets that has values as string and not meaningful after conversion to integer.

After steps of deleting some of less meaningful of datasets I decide to do some conversion with “case_status” datasets in order to model fitting by KNIME application that require that prediction dataset to be string. In addition, I do some conversion to some other datasets as well such as “case_received_date” and “decision_date” in to date and time then convert again to unix epoch numeric number, because the problem has occurred when I try to use any date datasets to fit model and the problem is these datasets currently are string and model accept only numeric data type.

On the other hand, I decide to fit model by using two different kind of application which are KNIME and Python libraries by using application called Jupyter application. As I mentioned before that KNIME need prediction dataset in string but in python they accept the current type of dataset of “case_status” without conversion, and the rest of datasets are using the same which are “case_received_date”, and “decision_date” but not limit to these two datasets, I also prepare some datasets to try and compare the result as well such as “foreign_worker_info_education” and datasets that has value as “Y” and “N” such as “job_info_alt_field”.

The data pre-processing and transformations you did (if any).

First of all, before apply data preprocessing technique to dataset I check all of datasets to see how many values has contain nothing, then I try to eliminate the entire rows of values. Mostly they have several missing values rows but for mainly dataset “case_received_date”, “decision_date” and “foreign_worker_info_education” are in quite perfect condition which is in “case_received_date” has only one missing value then I delete that entire row and the result my total dataset has contain only 1 less data value than original file. To check about datasets information, I use python library called pandas as shown below (Figure 1).

```
df = pandas.read_csv("new/TrainingSet(dateAndEducation).csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 147229 entries, 0 to 147228
Data columns (total 9 columns):
case_received_date               147229 non-null int64
decision_date                    147229 non-null int64
foreign_education_High School    147229 non-null int64
foreign_education_Associate's    147229 non-null int64
foreign_ducation_Bachelor's      147229 non-null int64
foreign_education_Master's       147229 non-null int64
foreign_education_Doctorate      147229 non-null int64
case_status                      147229 non-null int64
row ID                           147229 non-null int64
dtypes: int64(9)
memory usage: 10.1 MB
```

Figure 1: Datasets information.

In general, for data pre-processing the idea is to convert any datasets values that I see it possible to enhance my model accuracy I convert them all to the int datatype, and “case_status” as string datatype for KNIME application. First of all, for KNIME I change int to string by “0” to “a”, “1” to “b”, “2” to “c”, “3” to “d”. For date datasets I do data pre-processing as convert them to date and time first by using python library called pandas library with code as shown in Figure 2.

```
df_train['case_received_date'] = pandas.to_datetime(df_train['case_received_date'], format='%d/%m/%Y')
df_train['decision_date'] = pandas.to_datetime(df_train['decision_date'], format='%d/%m/%Y')
```

Figure 2: Conversion of date datasets as string datatype to date datatype.

After date datasets were converted I applied another code from same library as shown in Figure 3 in order to get value of date by convert them to epoch unix time as numeric data type (Figure 4).

```
df_train["case_received_date"] = pandas.to_numeric(df_train["case_received_date"])/1000000000
df_train["decision_date"] = pandas.to_numeric(df_train["decision_date"])/1000000000
```

Figure 3: Conversion of date datatype to numeric.

	unix_case_received_date	unix_decision_date	unix_pw_determ_date	unix_pw_expire_date	unix_recr_info_first_ad_start	unix_recr_info_second_ad_start	unix_recr_info_swa_jo
0	1446163200	1458259200	1458259200	1458259200	1458259200	1458259200	
1	1462406400	1471219200	1471219200	1471219200	1471219200	1471219200	
2	1404086400	1440633600	1440633600	1440633600	1440633600	1440633600	
3	1410220800	1422921600	1422921600	1422921600	1422921600	1422921600	
4	1433203200	1449187200	1449187200	1449187200	1449187200	1449187200	
5	1469491200	1476316800	1476316800	1476316800	1476316800	1476316800	
6	1467936000	1468540800	1468540800	1468540800	1468540800	1468540800	
7	1390953600	1434499200	1434499200	1434499200	1434499200	1434499200	
8	1439510400	1454889600	1454889600	1454889600	1454889600	1454889600	
9	1409616000	1424995200	1424995200	1424995200	1424995200	1424995200	
10	1375142400	1447286400	1447286400	1447286400	1447286400	1447286400	

Figure 4: Example of date conversion result.

Moreover, another data preprocessing technique that is used in this assignment is called binarization that convert data value in to “1” or “0”. This technique is used with several datasets which are “foreign_worker_info_education” by using code shown in Figure 5 and datasets that has values as “Y” to “1” and “N” to “0” (Figure 6).

```
# foreign_worker_info_education
df_test["foreign_education_High School"] = df_test.foreign_worker_info_education.map(lambda s: '1' if s == "High School" else 0)
df_test["foreign_education_Associate's"] = df_test.foreign_worker_info_education.map(lambda s: '1' if s == "Associate's" else 0)
df_test["foreign_ducation_Bachelor's"] = df_test.foreign_worker_info_education.map(lambda s: '1' if s == "Bachelor's" else 0)
df_test["foreign_education_Master's"] = df_test.foreign_worker_info_education.map(lambda s: '1' if s == "Master's" else 0)
df_test["foreign_education_Doctorate"] = df_test.foreign_worker_info_education.map(lambda s: '1' if s == "Doctorate" else 0)
```

Figure 5: Conversion of “foreign_worker_info_education” dataset to binary data type.

foreign_worker_info_education	foreign_education_High School	foreign_education_Associate's	foreign_ducation_Bachelor's	foreign_education_Master's	foreign_education_Doctorate
Master's	0	0	0	1	0
Bachelor's	0	0	1	0	0
Master's	0	0	0	1	0
Bachelor's	0	0	1	0	0
Master's	0	0	0	1	0
Doctorate	0	0	0	0	1
Master's	0	0	0	1	0
Master's	0	0	0	1	0
Bachelor's	0	0	1	0	0
Master's	0	0	0	1	0

Figure 6: Example of binarization technique result.

How you went about the problem

Normally, everything went quite well just only some small problems that has occurred in assignment process such as I have tried convert “case_status” from numeric to string by using excel application but it does not seem work well as it always convert back to numeric when I try to import the csv file into jupyter application so I leave it as numeric for python and also convert to “a-d” string as mention before in order to train model in KNIME application and it work perfectly fine. Another problem is when I process model training in KNIME application with my converted datasets even it has less datasets than original csv file but the performance of KNIME application has significant amount of time when comparing to Jupyter application which is KNIME application training time is around four to five hours but in Jupyter application use only about half of KNIME processing time. In addition, I spent sometimes with trying which application and datasets provide more efficiency of performance and some problems about coding error occurred several times, but it was my error that just small mistake about coding and naming of variable and files.

Classification techniques used and summary of the results and parameter settings

Figure 7 represent the coding of three steps which are firstly separate training datasets into two groups first group for training (80%) and second group for testing (20%) models. Second part is to apply datasets to training with different model from scikit-learn libraries to see which model is provide the best accuracy. As a result Figure 7 represent that the first place is KNN model with the most number which is 0.958980, second place is CART with 0.953056, and for what my assigned classifier base on classifier allocation from Classifier_Allocation.pdf file is SVM which is provide result only 0.883566 and it is consumed the most time of processing but still less than KNIME application.

```
# Split-out validation dataset
array = df.values
X = array[:,0:2]
Y = array[:,2]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size=validation_size, random_state=see
```

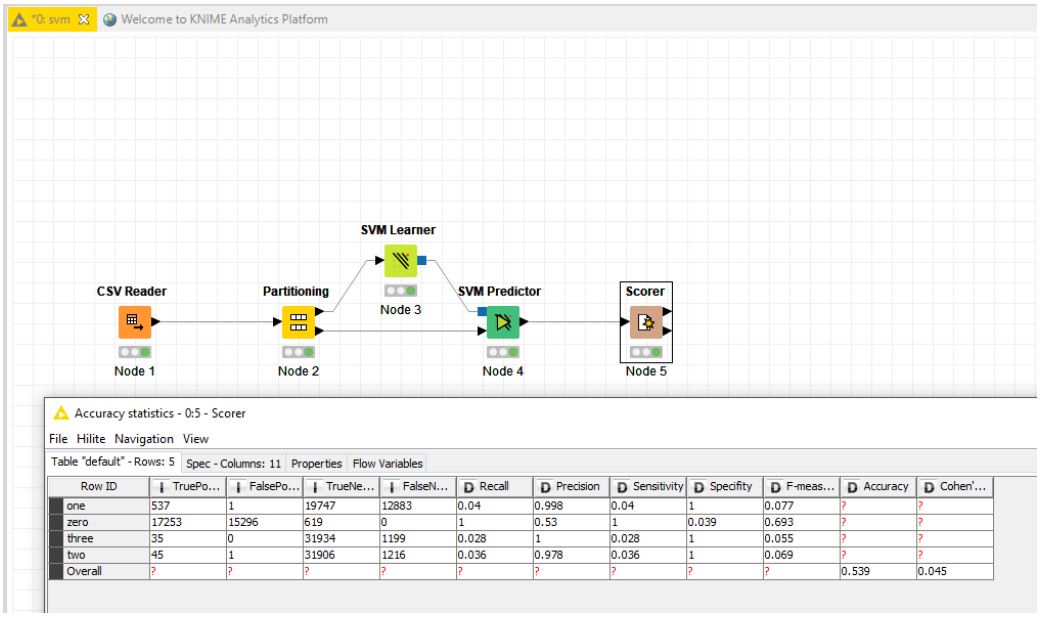
```
# Spot Check Algorithms
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.519928 (0.003221)
LDA: 0.537294 (0.004112)
KNN: 0.958980 (0.001917)
CART: 0.953056 (0.002411)
NB: 0.593510 (0.004497)
SVM: 0.883566 (0.003266)
```

Figure 7: Example of model testing results.

In addition, for KNIME application is used for testing only my main classifier which is SVM and the result is represent below (Figure 8) with accuracy result just 0.539 which is a little bit

different dataset from above and for kernel and parameters setting for this process is RBF kernel and sigma 0.1 because Polynomial and Hyper Tangent has an error occurred during training process.



Row ID	TruePo...	FalsePo...	TrueNe...	FalseN...	Recall	Precision	Sensitivity	Specifity	F-meas...	Accuracy	Cohen'...
one	537	1	19747	12883	0.04	0.998	0.04	1	0.077	?	?
zero	17253	15296	619	0	1	0.53	1	0.039	0.693	?	?
three	35	0	31934	1199	0.028	1	0.028	1	0.055	?	?
two	45	1	31906	1216	0.036	0.978	0.036	1	0.069	?	?
Overall	?	?	?	?	?	?	?	?	?	0.539	0.045

Figure 8: Example of training model result in KNIME application.

The actual classifier that you selected

According to model testing stage that represent KNN model has the highest accuracy rate and second place is CART model but after several tries of training with these different models KNN models represent the most reliable and stable accuracy rate, but CART and the rest models are provides sensitive result when compare to KNN model.

On the other hand, SVM model always represent accuracy rate that lower than 0.90s which is KNN is provide more dependable result. As a result, for this assignment I decide to train model with final dataset by using KNN model. In addition, for TestingSet.csv file I have cleaned and applied data pre-processing technique the same way as applied to training datasets. Figure 9 represent example of code that was used in prediction process.

```
#loading my train dataset into python
train_df = df
test_df = pandas.read_csv("new/TestingSet(dateAndEducation2).csv")

#factors that will predict the price
desired_factors = ['case_received_date','decision_date',"foreign_education_High School","foreign_education_Associate's",
                   "foreign_ducation_Bachelor's","foreign_education_Master's","foreign_education_Doctorate"
                  ]

#set my model to KNeighborsClassifier
model = KNeighborsClassifier()

#set prediction data to factors that will predict, and set target to SalePrice
train_data = train_df[desired_factors]
test_data = test_df[desired_factors]
target = train_df.case_status

#fitting model with prediction data and telling it my target
model.fit(train_data, target)

test_df['case_status'] = model.predict(test_data)
```

```
test_df.to_csv("new/resultTestingSet(dateAndEducation2).csv",index=False,encoding='utf-8')
```

Figure 9: Example of prediction process.

Reflection section

For the most valuable things that I have learned in this assignment 3 is the most important part is datasets which is required the most time to cleaning, considering meaningful datasets, and the accuracy is depending on all stage that relate to datasets. The second thing is the right tool provides dramatically different of result and resource consumption such as if I use python it has a lot more tutorial and online resource to learn how to use it. On the other hand, KNIME is a lot easier to use which suitable for who do not know about coding, but the disadvantage is the lower performance than python. Another thing is about in my opinion I have put an effort in this assignment quite a lot because it consumes and require load of times in order to understand various type of things such as datasets, classifier methodology, implementation with application, and etc. which is the most of the time I have spent with this assignment is going through the wrong way first but it was really helpful for me after I dive deep down through the wrong way, it made me more comfortable with everything slowly and when I found the way I thought I would be the correct way I can do it with my really understanding which might not clearly and gain full knowledge but I feel that I can do it quicker than first ten or twenty hours of working.

In conclusion, if I would have to do it again I believe that I would do it more efficiency and productive because now I obtained some knowledge about how I would deal with the datasets and which one of them is not useful and meaningful after conversion or applied data pre-processing technique that I believe it will be consume less time as before. The next thing is that I have familiarity with coding in Python and I believe that I could do it more comfortable next time because this coding process is also consumed a lot of my time to learn and understand what code is produce what I want such as cleaning data, drop table, present information, describe dataset and etc. lastly, I also have an idea that I should have learned more about methodology and formula carefully because this would make a big different if I have a clearly of knowledge in everything so I believe I can adapt and using it in order to enhance the accuracy in each models.