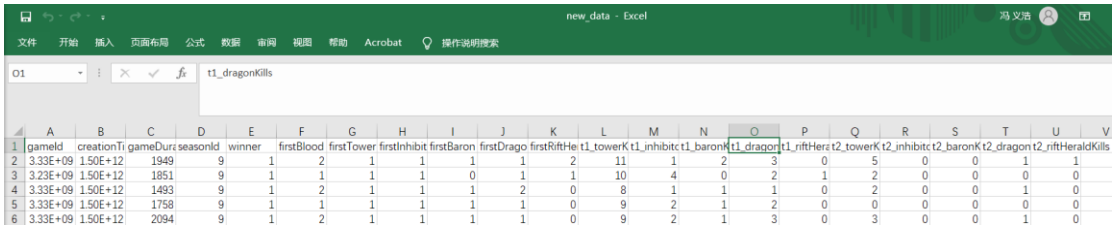
1 Introduction

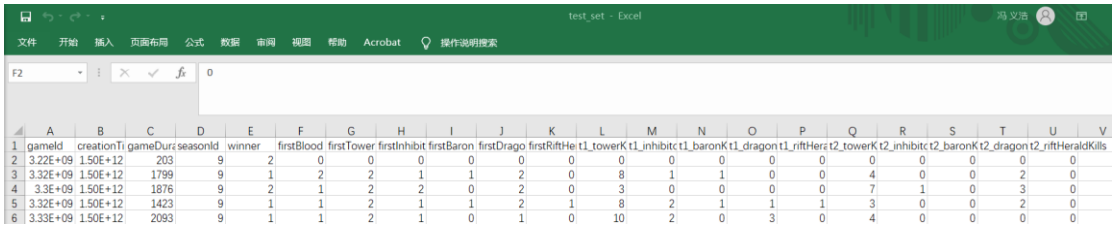
The project is aimed irrefutably at guiding us to have a more skilled command of what we have learned during several weeks' studying. In order to achieve this teaching target, we are given an unambiguous topic on a famous game, which is called League of Legends. The project supplies us with two files which preserve a large amount of game records from game environment description to performances of both teams. Under the circumstance that we have a crazy interest on predicting which team will win the final champion after a long period of fierce fighting, we are asked to evaluate and make comparisons among the accuracy of the models that we have trained without restrictions by our freely chosen features that are extracted from dataset.csv file. The pros and cons of different models would be observed by having a clear look on both the calculating results and some related diagrams.

As for the methodology, I thought it covers the basic concepts of efficient data mining as well as appropriate data processing. The major ideas of measures I take in this project is to choose several common classifiers and optimize each evaluating values by changing and adjusting parameters.

Note that, what follows is a short piece of two files, I hope they can help you build a rough impression on what I would introduce next and get to know where the accuracy comes from(column E will be focused).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	gameId	creationTi	gameDura	seasonId	winner	firstBlood	firstTower	firstInhibit	firstBaron	firstDrago	firstRiftHe	t1_towerK	t1_inhibitc	t1_baronK	t1_dragon	t1_riftHera	t2_towerK	t2_inhibitc	t2_baronK	t2_dragon	t2_riftHeraldKills	
2	3.33E+09	1.50E+12	1949	9	1	2	1	1	1	1	2	11	1	2	3	0	5	0	0	1	1	
3	3.23E+09	1.50E+12	1851	9	1	1	1	1	0	1	1	10	4	0	2	1	2	0	0	0	0	
4	3.33E+09	1.50E+12	1493	9	1	2	1	1	1	2	0	8	1	1	1	0	2	0	0	1	0	
5	3.33E+09	1.50E+12	1758	9	1	1	1	1	1	1	0	9	2	1	2	0	0	0	0	0	0	
6	3.33E+09	1.50E+12	2094	9	1	2	1	1	1	1	0	9	2	1	3	0	3	0	0	1	0	



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	gameId	creationTi	gameDura	seasonId	winner	firstBlood	firstTower	firstInhibit	firstBaron	firstDrago	firstRiftHe	t1_towerK	t1_inhibitc	t1_baronK	t1_dragon	t1_riftHera	t2_towerK	t2_inhibitc	t2_baronK	t2_dragon	t2_riftHeraldKills	
2	3.22E+09	1.50E+12	203	9	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	3.32E+09	1.50E+12	1799	9	1	2	2	1	1	2	0	8	1	1	0	0	4	0	0	2	0	
4	3.3E+09	1.50E+12	1876	9	2	1	2	2	0	2	0	3	0	0	0	0	7	1	0	3	0	
5	3.32E+09	1.50E+12	1423	9	1	1	2	1	1	2	1	8	2	1	1	1	3	0	0	2	0	
6	3.33E+09	1.50E+12	2093	9	1	1	2	1	0	1	0	10	2	0	3	0	4	0	0	0	0	

2 Algorithms

In this project, I choose three classifiers that are Decision Tree Classifier, Support

Vector Machines and K-Nearest Neighbor. I suppose the first classifying method is most familiar to people. Hence, I will introduce them according to the order I have mentioned above.

2.1.1 Decision Tree can achieve the target of getting the probability of the mathematical expectation of net present value no less than zero by constructing a tree. In general, we would have collected a set of all attributes and have known all the probabilities of each condition happening. Thanks to these data, if you make your decision tree model visualized, it would be a flowchart having tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each of leaf node(terminal node) holds a class label. A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general, decision tree classifier has good accuracy. Hence, decision tree induction is a typical inductive approach to learn knowledge on classification.

2.1.2 Support Vector Machine (SVM) is a relatively simple Supervised Machine Learning Algorithm used for classification and regression. It is more preferred for classification but is sometimes very useful for regression as well. Basically, SVM finds a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line. In SVM, we plot each data item in the dataset in an N-dimensional space, where N is the number of features or attributes in the data. Next, find the optimal hyperplane to separate the data. So by this, you must have understood that SVM can only perform binary classification (i.e., choose between two classes) inherently. However, there are various techniques to use for multi-class problems. We use Kernelized SVM for non-linearly separable data. That is to say, we have some non-linearly separable data in one dimension. We can transform those data into two-dimensions and the data will become linearly separable in two dimensions.

This is done by mapping each 1-D data point to a corresponding 2-D ordered pair. So for any non-linearly separable data in any dimension, we can just map the data to a higher dimension and then make it linearly separable. This is a very powerful and general transformation. Worth to mention that, a kernel is nothing a measure of similarity between data points. The kernel function in a kernelized SVM tell you, that given two data points in the original feature space, what the similarity is between the points in the newly transformed feature space. More information on kernel will be given near the bottom of the algorithms.

2.1.3 K-Nearest Neighbors is also one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection. It is widely disposable in real-life scenarios since it is non-parametric which means that it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). In general, there are two properties that define KNN well. Lazy learning algorithm − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification. Non-parametric learning algorithm − KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

2.2 I have introduced my chosen classification methods above, here I will present some information about the parameters inside the algorithms and show you their pros and cons according to the same order.

2.2.1 Let' s begin with Decision Tree. Quite a lot parameters exist in this algorithm, however, I found out that only max_depth can have an apparent impact on the result of accuracy. Other parameters seem to be weak in this case of project 1, like criterion which we could choose either gini or entropy. Hence, the only powerful one is max_depth. It is a kind of data of integer attribute and it describes the maximum depth of the tree just as what the name is called. The value defaulted is set to be "None", and then the nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. How should we understand the change on the value of

accuracy resulting from adjustments on max_depth? Think, if we can control the maximum depth of the tree, imagine that some terminal nodes would have stopped before they meet purity. Indeed, it is not necessary to split the nodes using all of the attributes and sometimes it is a waste of time and resource to go deeper. Hence, figuring out an appropriate depth plays a significant role at this stage of predicting and evaluating the final outcomes.

2.2.2 As for the Support Vector Machine, I put my focus on "kernel" and "gamma". Obviously, there are various kernel functions available, but two of them are very popular. One is Radial Basis Function Kernel (RBF), the similarity between two points in the transformed feature space is an exponentially decaying function of the distance between the vectors and the original input space as shown below.

$$K(x, x') = exp(-\gamma||x - x'||)$$

Note that RBF is the default kernel used in SVM. The other one is called Polynomial Kernel which takes an additional parameter, "degree" that controls the model' s complexity and computational cost of the transformation. But in this project, I set the kernel from a domain of "linear, rbf, sigmoid". By applying different kernels, I finally choose rbf for its accuracy is highest. One more parameter is "gamma", this parameter decides how far the influence of a single training example reaches during transformation, which in turn affects how tightly the decision boundaries end up surrounding points in the input space. If there is a small value of gamma, points farther apart are considered similar. So more points are grouped together and have smoother decision boundaries (may be less accurate). Larger values of gamma cause points to be closer together (may cause overfitting).

2.2.3 In K-Nearest Neighbors, the value of K is of most importance. In order to assist you to understand, I first show you a graph below.

Step 1 − For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

Step 2 − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

Step 3 − For each point in the test data do the following −

- **3.1** − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.
- **3.2** − Now, based on the distance value, sort them in ascending order.
- **3.3** − Next, it will choose the top K rows from the sorted array.
- **3.4** − Now, it will assign a class to the test point based on most frequent class of these rows.

Step 4 − End

In short, this kind of algorithm can optimize the evaluation by changing the value of K. This leads us to determine a proper range and finally find the best K after several trials.

2.3 A few graphs is given to present **pros and cons of the three classification methods** above.

2.3.1 Decision Tree.

Strengths	Weakness
Decision trees are able to generate understandable rules.	Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
Decision trees perform classification without requiring much computation.	Decision trees are prone to errors in classification problems with many class and relatively small number of training examples.
Decision trees are able to handle both continuous and categorical variables.	Decision tree can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be

	found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.
Decision trees provide a clear indication of which fields are most important for prediction or classification.	

2.3.2 Support Vector Machine.

Strengths	Weakness
They perform very well on a range of datasets.	Efficiency (running time and memory usage) decreases as size of training set increases.
They are versatile : different kernel functions can be specified, or custom kernels can also be defined for specific datatypes.	Needs careful normalization of input data and parameter tuning.
They work well for both high and low dimensional data.	Does not provide direct probability estimator.
	Difficult to interpret why a prediction was made.

2.3.3 K-Nearest Neighbors.

Strengths	Weakness
It is very simple algorithm to understand and interpret.	It is computationally a bit expensive algorithm because it stores all the training data.
It is very useful for nonlinear data because there is no assumption about	High memory storage required as compared to other supervised learning

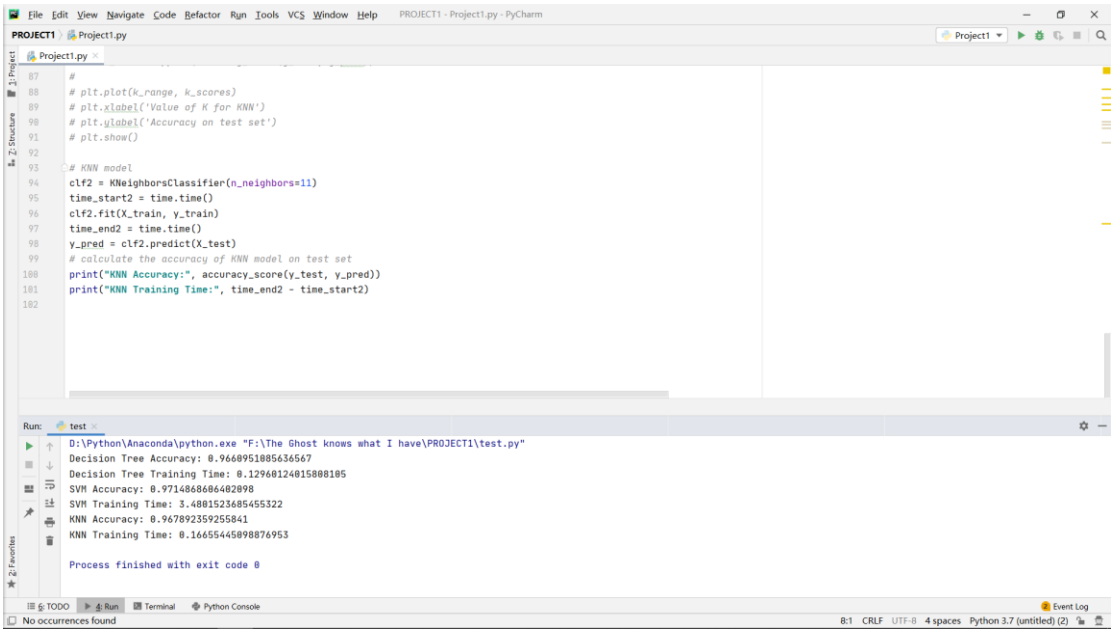
data in this algorithm.	algorithms.
It is a versatile algorithm as we can use it for classification as well as regression.	Prediction is slow in case of big N.
It has relatively high accuracy but there are much better supervised learning models than KNN.	It is very sensitive to the scale of data as well as irrelevant features.

3 Requirements

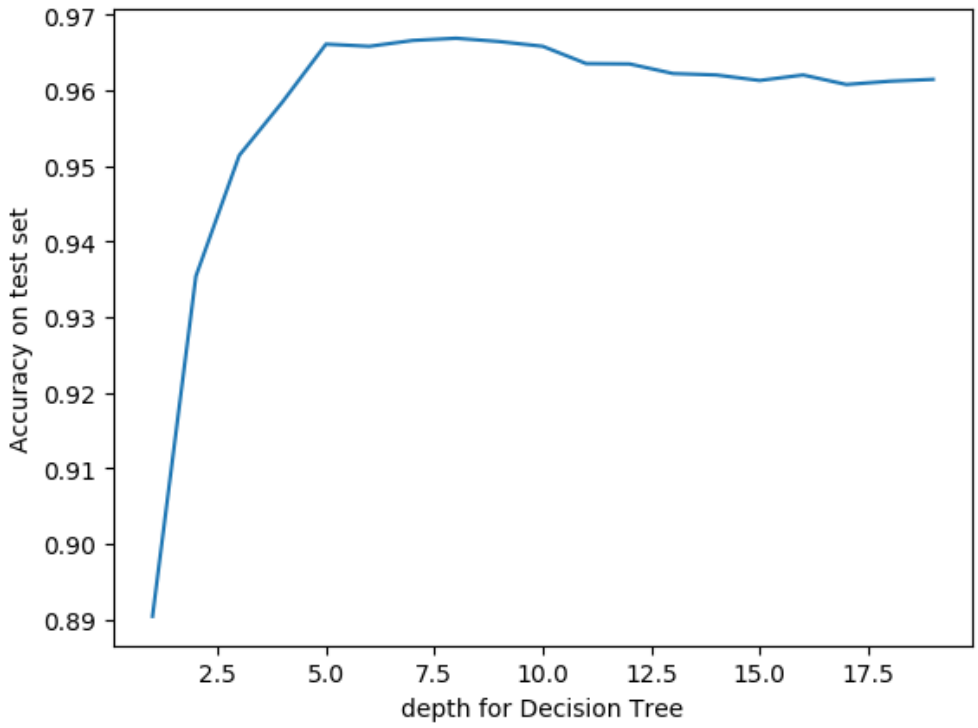
From	Import
	pandas
	numpy
	matplotlib.pyplot
sklearn.metrics	accuracy_score
sklearn.tree	DecisionTreeClassifier
sklearn.model_selection	train_test_split
sklearn.model_selection	GridSearchCV
sklearn.svm	SVC
sklearn.neighbors	KNeighborsClassifier

4 Results

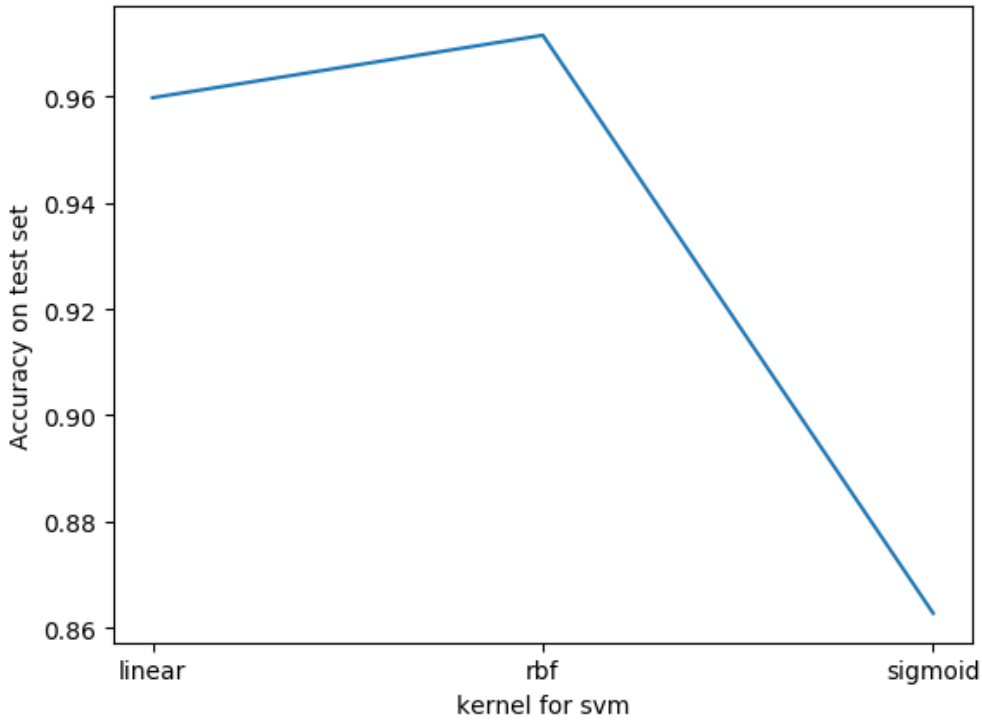
Classification	Accuracy	Training Time
Decision Tree	0.9661	0.1296
Support Vector Machine	0.9715	3.4802
K-Nearest Neighbors	0.9679	0.1666



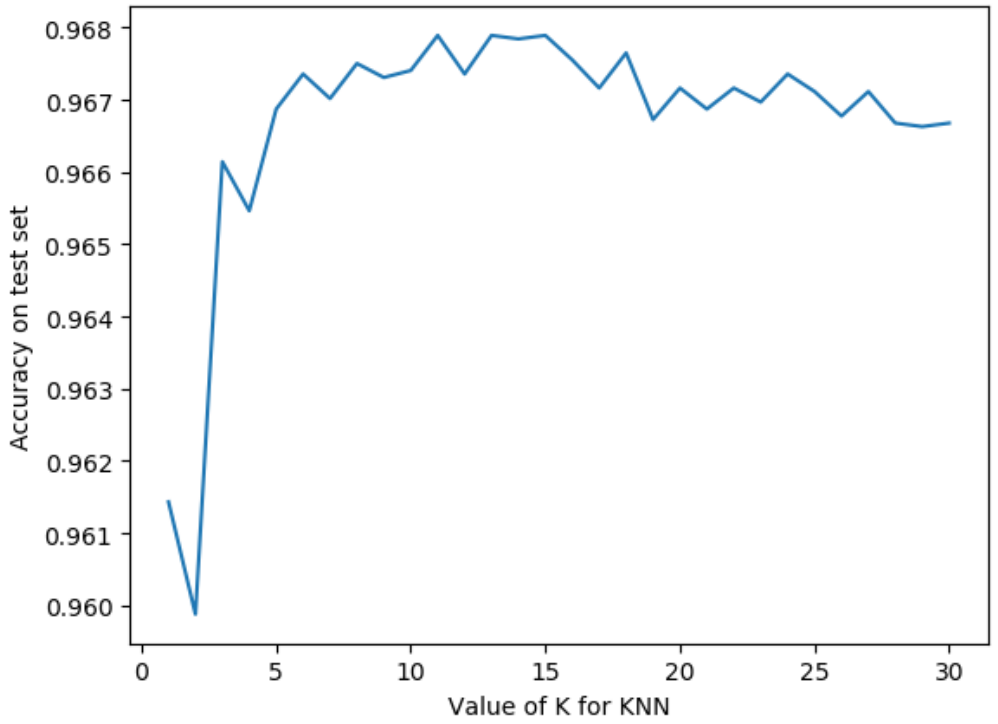
If you want to touch **how I make the adjustments for these important parameters**, here are three diagrams I prepared to help you understand. Also, they are really powerful evidence of what I have mentioned in the algorithms(i.e., set a domain of values or a range of different parts, plot and observe the condition which guarantees best outputs.)



Best max_depth is 5.0.



Best kernel is rbf.



Best K is 11.

5 Comparison and discussion

Even though I have received each lecture carefully and try my best to review all the teaching materials, I find out that if one wants to understand what we have learned during these several weeks, there exists a lot of space to dig out. Through this project, it makes me return to the original lessons again and again. Moreover, it is still a big challenge to surf on the pure English websites to search for the useful information I need urgently. Sometimes, reading an article of particular topic on classification costs me quantities of time since I cannot fully understand it or pick necessary assistance if I read it only once. Certainly, it is of surprise to have a look on different classification methods and the course of choosing the methods and make adjustments does make this project unforgettable. However, what bothers me most is an embarrassing reality that I have never ever played this game. As a result, there are a lot of concepts makes me confused and I have to spend more time than those students who have enjoyed LOL getting knowledge from here and there in order that I could imagine what the game is.

In my code, the parameters have been optimized in a way. For example, I plot the curve of accuracy to the depth of tree by matplotlib.pyplot library. Although I could determine the model reaches the biggest accuracy under the condition that the value of depth equals to five, I have no idea whether there exists more higher accuracy or not if I attempt to choose other combination of attributes and set values for more parameters in the decision tree framework. If more time is allowed, the pity above may have a chance to erase. I would also have the courage to give ensemble classifiers a go.

Go back to the results, the best accuracy is reached by SVM even though it costs the longest time owing to the characteristics of Support Vector Machine classifier. As a matter of fact, other two models predict the labels quite perfectly as well. All of them could be higher than 0.96 which means that only one error could happen in a total prediction of 25 samples of LOL competitions.

In a word, all of the experience makes learning this subject meaningful. Through this project, I have a better command of data mining and processing. Hopefully, a broader world will present in front of my eyes and studying never stops.