Project in Computer Science EDAN70 - Machine Learning

Julian Cieplik, Fredrik Sydvart
dat12jci@student.lth.se, dat12fsy@student.lth.se

Abstract—**Project in Computer Science course EDAN70 at Lunds Tekniska Högskola. The project is part of artificial intelligence and we wanted to get a deeper understanding of the subarea machine learning and how it can be applied to real world data. The focus of the project is on a problem to classify hotel recommendations by using different machine learning models and also presenting a work flow for iteratively improving the machine learning model.**

Index Terms—**Machine learning, MINST, Random Forest Classifier, C-Support Vector Classification (SVC), Cross-validation, Scikit-learn, Feature engineering**

I. Introduction

The goal of this project is to get a deeper knowledge of a field in artificial intelligence. We chose to focus on applied machine learning where we have data sets from real world problems and apply different models to train and predict data.

Kaggle.com is a website where users from all over the world can compete to produce the best machine learning models. The website provides all kinds of interesting datasets to work with during competitions and also some for learning in a non-competitive environment. Users range from new amateur programmers to PhD students. Many organizations don't have access to advanced machine learning or perhaps they want new insight on how to build a better model that provides the maximum predictive power for their data. Kaggle offers companies a cost-effective way to solve their problems and at the same time give data scientists and statisticians real-world data in order to develop their techniques and win prize money.

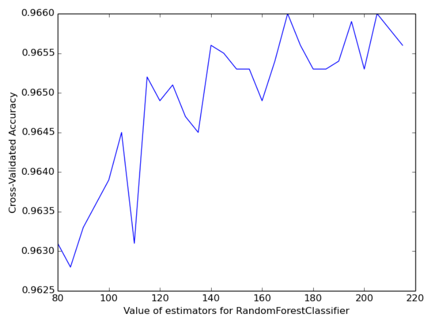
To get familiar with machine learning libraries and different strategies to solve problems, we started with an introductinary problem - Digit Recognizer on Kaggle. The goal was to later utilize our knowledge to compete on Kaggle for the Expedia Hotel Recommendations competition and iteratively climb the leaderboard by applying various machine learning algorithms and evaluating their trained classification models.

II. Digit recognizer

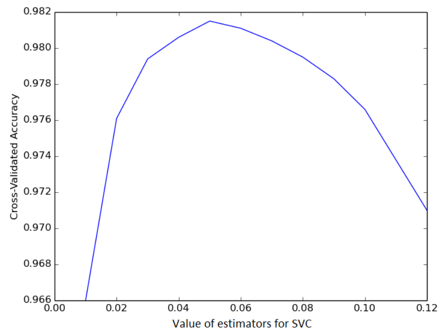
This problem was an introductionary dataset for machine learning with Kaggle. The data is taken from the MNIST ("Modified National Institute of Standards and Technology") dataset which is a classic within the machine learning field and have been extensively studied. We were given a training and test set both containing a handwritten single digit in each image. The images had 28 pixel height and 28 pixel width and the digit was a number between 1-9. The goal is to determine what the digit is. Scores are computed through how many handwritten numbers where correctly classified of the total evaluation.

Our initial solution was to use library called Scikit-learn which is a simple and efficient tool for data mining and data analysis written in Python. The first approach was to use Random Forest classifier and with this algorithm we scored 0.96529. The score means that we correctly classified approximately $96,5\%$ of the digits. The next approach was to use C-Support Vector Classification (SVC) from the Scikit-learn tool library in which we scored 0.98243. By changing the gamma value 0.05 for the SVC we managed to improve the score to 0.98371.

After trying some different classifiers we hit the maximum submission limit on Kaggle which lead to us having to evaluate the result ourselves. We decided to use cross-fold validation on each classifier method for different parameters in order to check their accuracy of predicting correctly and use this when optimizing our result. We luckily found that the best classifier and parameter was the SVC with the gamma value 0.05, which we already have tested on Kaggle.



(a) Random Forest classification with number of estimators. More parameters could possibly be needed since the trend has not converted yet.



(b) Support Vector Classification (SVC) with gamma values.

Fig. 1: Different classifier and parameter testing with cross-fold validation.

III. Expedia

Expedia is an online travel company which provides hotel reservations, car rentals and flight booking through their website. Expedia has issued a competition to answer the question of which hotel type will an Expedia customer book? The problem to solve for the provided data set is to determine ids for clusters of hotel recommendations given some search meta data collected from Expedias user base. The data set have clearly defined columns, it is easy to understand what kind of data we are dealing with because of the non-cryptic column names, making this competition ideal for us. Additionally, the data is made anonymously, meaning all the search queries has been converted to an integer value, for example if the user searches for a specific destination, then we won't have the name of the place but rather an integer value. Expedia is challenging users on Kaggle to provide context to each search done by a customer and predict the likelihood that the user will stay at one of the 100 predefined different hotel groupings. Each grouping or cluster has a similar pattern of searches landing in it. In other words, hotels matching a search query (based on historical price, customer star ratings, geographical locations relative to city center, etc) are put together in a matching cluster. For new searches the model should predict in which hotel cluster the search example is most likely to belong to. [2] [3]

A. Data contextualized

In order to try a machine learning model it is a good idea to understand the data we are working with. The information contained in the dataset describes how a user made a search for a hotel. See appendix for information stored in the data. The parts which make up each search can be for example if the user desires a package deal where a flight is included, the country the traveler wishes to go to or, dates of checkin and checkout, and number of companions. There are several data fields arranged in columns which make up the whole dataset. One search example corresponds to a row which has 24 different queried features and values for those features. There are 37 million examples in the training set and 2 million in the test set. Additionally, we are supplied with a destination file which is just one additional feature for a search. It carries latent information about the search destination. Its safe to assume that its some combination of hidden destination characteristics which have been inferred from observation of other variables. At the same time the training set has the target as the last column, which is the hotel cluster id.

To understand the data better we take a look at the Expedia website shown in figure 2 and explain it:

- *Going to*: fills in the columns srch_destination_type_id, hotel_continent, hotel_country, and hotel_market
- *Check in/out*: maps srch_ci and srch_co respectively
- *Guests*: determines the number of guest in fields srch_adults_cnt, srch_children_cnt, and srch_rm_cnt (rooms)
- *Add a flight*: maps to the is_package field

After pressing the "Search" button we are transferred to a new web page where there are several other fields for filtering out the hotels:

- *Hotel class*: 5 to 1 star(s)
- *Price Per Night*: 5 price limits
- *Search Nearby*: different attractions nearby
- *Property Type*: Hotel, Guest house, etc

These could be the characterization of a hotel cluster. The problem becomes clear now. We are asked to predict from a given search the best hotel clusters. Therefore, hotel clusters could be some kind of set of attributes which are most likely to fit the search, such as the filtering in the list above.

The classification is useful for Expedia because they can recommend hotels at a much earlier stage of the visit to Expedia.com and also limit the number of displayed hotels. The problem e-commerce is experiencing during the earlier stages of the conversion funnel, before a user has filled in all the boxes and clicked on all the buttons, is that there is not

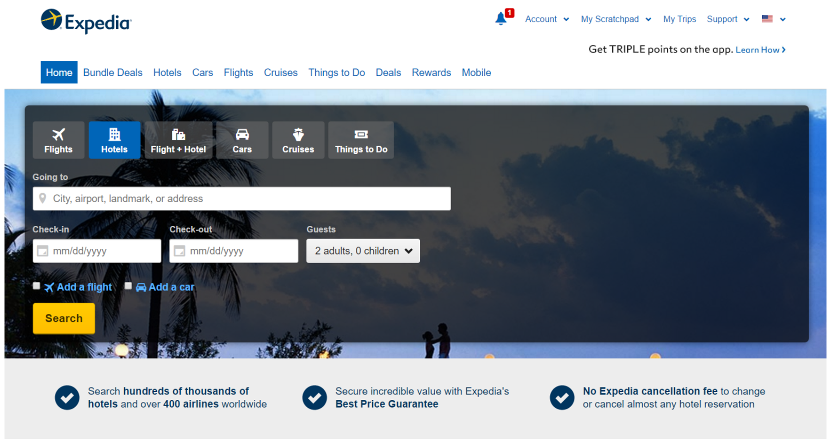

Expedia
Account
My Scratchpad
My Trips
Support
Get TRIPLE points on the app. Learn How
Home
Bundle Deals
Hotels
Cars
Flights
Cruises
Things to Do
Deals
Rewards
Mobile
Flights
Hotels
Flight + Hotel
Cars
Cruises
Things to Do
Going to
City, airport, landmark, or address
Check-in
mm/dd/yyyy
Check-out
mm/dd/yyyy
Guests
2 adults, 0 children
Add a flight
Add a car
Search
Search hundreds of thousands of hotels and over 400 airlines worldwide
Secure incredible value with Expedia's Best Price Guarantee
No Expedia cancellation fee to change or cancel almost any hotel reservation

Fig. 2: Expedia

much information to rely on. The classification can thus serve as an pre-filtering step for narrowing the recommendations by defining the span of the search in respective hotel cluster. Therefore, more information is extracted from the early data. Later on an in-house algorithm can find matching hotels for the cluster and sort them according to price, accommodation, etc.

> "During a user visit, and prior to any clicks, we can show hotels that are likely to be booked on top of hotel sort, on expedia home page, etc. We can also include hotel recommendation in emails we send to customers." - Kaggle Competition Admin Adam on a forum topic *How useful will these recommendations be?* [1]

B. Evaluation

Submissions on the competition are evaluated according to the Mean Average Precision @ 5 (MAP@5) function:

$$MAP@5 = \frac{1}{|U|} \sum_{u=1}^{|U|} \sum_{k=1}^{min(5,n)} P(k)$$

where $|U|$ is the number of user events, $P(k)$ is the precision at cutoff k and n is the number of predicted hotel clusters. A higher score is given the closer the prediction are to the correct hotel cluster id. If the correct cluster id is not within the top 5 predicted ids, the score is then 0. When a submission has been submitted and a score calculated, a public score is then posted on the leaderboard on Kaggle. [4]

C. Random Forest Classifier

The first machine learning model we tried in the competition was the Random Forest classifier. Random forest is a meta estimator and an ensemble learner. that constructs a multitude of random decision trees at training stage from various sub-samples of the dataset and outputs the class that is the mode of the classification classes for each individual tree. Mode can best be described as the most common value in a given series of values. Random forest is still one of the strongest supervised learning methods and is also one of bagging methods.

Each decision tree is based on random selection of training data. The data can be part of one or more subset used in generating decision trees. As the trees are based on random selection of data, each tree can be called a random tree. By combining multiple trees we get a forest, hence the name Random Forest classifier.

Since its weak classifier is a decorrelated decision tree, the correlation between subsets is decreased and together with averaging the result of each decision tree, it helps to improve the predictive accuracy and control over-fitting. The method is also relatively easy to implement, making it a favorite among engineers to even implement it from scratch if needed.

Figure 3 explains well how Random Forest most important step works. The algorithm generates a lot of weak decision trees and their outputs are converged with a certain prefixed rule.

A strong part of Random Forest is that it corrects for the habit of over-fitting to their training set for each decision tree.

In order for the method to work we have to make two beliefs:

1) Most of the tree can provide correct prediction of class for most part of the data.
2) Each individual tree make mistake at different place. One tree might be correct where another tree is not.

From this we get that the method has to conduct voting for each of the observation for value and then decide about the tree to use based on the result. [5] [6] [7]

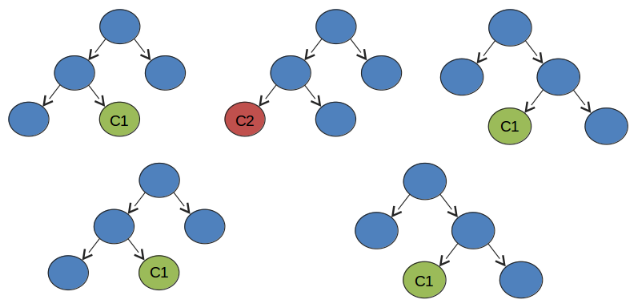


Fig. 3: We can see that the tree predicting C2 is probably making a mistake, since the rest of trees are predicting another value C1. Together with the assumption that a tree mostly provide correct prediction for the most part of the data, the classifier can conclude that the correct prediction in this case probably is C1.

D. N-fold Cross-validation - Estimating performance of a model

After hitting the daily allowed submitted submissions on Kaggle we needed a better way to test our model. The idea with cross-validation is to have a model validation technique which can be used to optimize a model easily by validating the model on different parameters. The cross-validation technique is mainly used where the goal is prediction and to assess how good the model will perform in practice.

Cross-validation is given a dataset of known data on which training is done. Then another unknown dataset or a part of the known data which have been withheld from training is used by the model to predict and give some kind of score on how good the model can perform and generalize over new data. This approach limits the problem of overfitting the model.

With N-fold cross-validation, data is divided up into n chunks and trained n times. Each chunk will be treated differently as the holdout set each time.

N-fold cross-validation is used for repeating the train and test split procedure over multiple times. For each trial of train and test split, there is a variance associated with it, and with N-fold cross-validation this can be used to reduce the variance. The entire dataset is split into N equal size ”folds”, and each fold is used once for testing the model and $N-1$ times for training the model. This process is shown in figure 4. [8]

E. Grid Search - Searching for estimator parameters

With Grid Search in the Scikit-learn library we can search for a parameter that is not directly learnt within the estimator model that optimizes for the best cross-validation estimator performance score. A typical example could be to search for the best number of estimators in the Random Forest Classifier. Any parameter that is provided when constructing an estimator in Scikit-learn may be optimized with grid search.

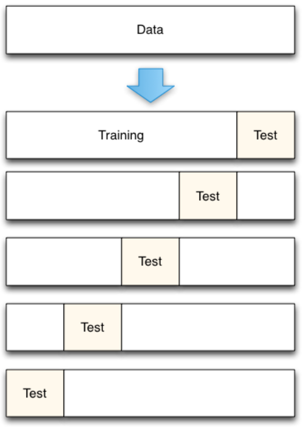


Fig. 4: Schematic figure over the principle of N-fold cross-validation. In this case $N = 5$ and the model is used independently on each fold.

As read from the documentation in the Scikit-learn, the grid search works by using multiple things:

- An estimator model (such as Random Forest Classifier, SVM or Gradient Boosting).
- A parameter space.
- A method for searching or sampling candidates.
- A cross-validation scheme.
- A score function. In our case the MAP@5 which we implemented in Python as a function.

We used grid search to find the best parameter for random forest, gradient boosting and SVM. In this way we could automate the procedure of finding a good model. Example of using the grid search for random forest classifier:

```
# n_estimators - the number of estimators
# n_jobs - number of jobs to run in
parallel
# max_depth - the maximum depth of the
tree
params = {'n_estimators': [150],
          'n_jobs': [2], 'max_depth': [4]}

# Create estimator model
randomforest = RandomForestClassifier()

# Create Grid Search with parameters and
# use cross folds as a cross-validation
method
# together with a custom scoring method
gridsearch = GridSearchCV(
    estimator=randomforest,
    param_grid=params, cv=CROSS_FOLDS,
    scoring=map5eval)
```

F. *Most popular local hotels*

After trying with some of our models we presented above, we read on the forum how some users found a pretty good method to get a good scoring in the competition. The method consisted of looking at different features and calculating the most popular local hotels. The result from this method will be presented in the result section. Later on an even more advanced version appeared of most popular local hotels which scores better.

G. *Data leakage*

During week 6 in this project, a data leakage was discovered and confirmed by the competition admins on the forums on Kaggle. The competition admin stated that the competition would continue without any change.[10]

Data leak is when unexpected additional information appears in the training data, which allows for a model to make unrealistically good predictions. The worst part is that the model will appear to be fully functional and working with all the data, but when put in a real world situation with new data, the model could potentially fail or give totally wrong predictions. Because of this, models over-present their generalization error and often render them useless in the real world. The effect for a data leakage can vary between human or computer error. Sometimes data leakage can even be intentional or unintentional.

In the competition, the leak had to do with the `orig_destination_distance` attribute. The competition admin stated that they estimate that the leak affects approximately $1/3$ rd of the holdout data. By using this leak you can find `hotel_clusters` for the affected rows by matching rows from the train dataset based on the following columns: `user_location_country`, `user_location_region`, `user_location_city`, `hotel_market` and `orig_destination_distance`. However, in the real world hotels can change cluster assignments since they depend on parameters such as hotel popularity or the price which varies with season. Since hotel cluster assignment can change, any model that utilize the leak will render useless in the real world.

Because of the leak, the competition has turned into using a model that utilize the leak for $1/3$rd of the data, and using a true machine learning model to predict the rest $2/3$ of the data.

IV. Result

Result are presented in table I.

V. Related Work

Kaggle has a good feature open for everyone, and that is something called Scripts. Users can run R and Python code directly on a dataset without needed to download the data or leave the browser. All scripts are open for everyone once it has been created. Each script can only be remotely run for a limited amount of time and resources. This enables everyone to try machine learning without the need for expensive hardware.

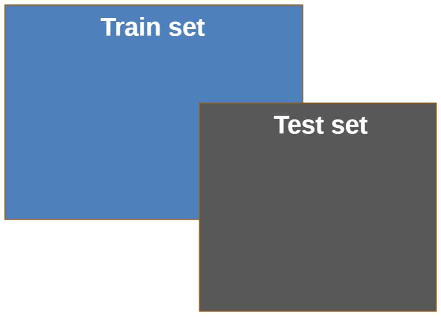


Fig. 5: Schematic figure over the concept of a data leak in machine learning.

TABLE I: Result with different models

Model	Score
Random Forest Classifier	0.18584
Most Popular Local Hotels	0.30090
Data Leakage with most popular local hotels	0.50050

We forked and improved scripts both for the initial most popular local hotels script, and also for the same but improved script that utilized the data leak.

The most popular local hotels approach is building a dictionary where the `srch_destination_id` is the key. A list of different `hotel_cluster` and their frequency in a tuple is the value. Every training sample is thus iterated over and it's `srch_destination_id` acts as key to the dictionary and the frequency for the targeted `hotel_cluster` is updated. Predicting is simply looking up each test samples `srch_destination_id` and taking the 5 most frequent `hotel_cluster`.[9]

VI. Conclusion

In this project we got a deeper understanding of machine learning and how it can be applied to real world data. We used a systematic work flow to climb the leaderboards and iteratively improve our models by using different techniques such as N-folds cross-validation and combining grid search with Scikit-learn python library.

The models that best worked for us was using a tuned random forest classifier on which we scored 0.18584 in the public leaderboard. It was quite surprising for us that most popular local hotels, which is as easy as using a simple non-machine learning approach by looking what most users book, could yield a score of 0.30090, which actually was better then our own model. The approach of using a most popular local hotels was not even a benchmark in the leaderboard, which could indicate that the method was not intuitive, even though it makes sense once we see the results since we have so much information regarding different users in the data. We don't think that Expedia would want a most popular local hotels implemented on their website, as it was stated in the *Data contextualized* section, e-commerce today do not have much information to rely on in the earlier booking stage. However, the most popular local hotels could probably be used as a complement to a machine learning method.

With more efficient computers and time we could have explored a better model to use for the rest of the holdout data that was not part of the data leak. That could have scored among the top. Another method could be to completely disregard the data leak and try to find an another model, however the best model in this competition would probably be a model that utilizes the data leak in some way.

Since the data leak was discovered, the competition was not ideal to explore the area machine learning. But we managed to still get a deeper understanding of practical machine learning using real world data and how it could be used in future products or services in the real world. Overall the competition was still favorable for the project despite the data leak.

VII. Project work flow

w.1 Created report on overleaf. We studied the top Kagglers for their methodologies in ranking high on the leaderboards.

w.2 The first submissions to kaggle.com for the Getting Started problem Digit Recognizer. We set up simple SVC and RandomForest learners and evaluated their performance using k-crossfolds.

w.3 Main competition - Expedia Hotel Recommendations. Investigated how the data was structured and which machine learning algorithm would suffice for an initial submission.

w.4 Implemented Random Forest classifier for the Expedia Hotel Recommendations competition. Made our initial submission and scored 0.18584 out of 1.0 and placed us at 351 out of 426 contestants as of the date for the submission (26/4). Our next step is to use cross-folds to find a good parameter for the Random Forest Classifier. After reading on some user submitted scripts on Kaggle, we found out that using a greedy most popular local hotels was very easy and popular to use. This approach was pure feature engineering and to check which `srch_destination_id`, ID of the destination where the hotel search was performed, that was most commonly used together with a `hotel_cluster`, ID of a hotel cluster. The output from the testing data was then the most popular hotel cluster ID for the specific `srch_estination_id`. This new approach yielded better results and a significant performance boost. The new score gave us 0.30090 and placed us at 242 out of 427 contestants (26/4). We improved our score with 0.11506 and moved up a total of 110 positions. The greedy approach can probably be improved with some additional feature engineering.

w.5-6 Project work mostly suspended because of exams. Started to investigate into the Expedia leak.

w.7 Submission of Expedia leak and updated our score in the leaderboard. Started to work on the presentation and trying to complete most of this report. Made our last submission with most popular local hotels together with utilizing the data leak and scored 0.50050.

w.8 Work mostly consisted of finish the presentation of this report.

References

[1] kaggle.com Competition Forum, Topic ”How useful will these recommendations be?” https://www.kaggle.com/c/expedia-hotel-recommendations/forums/t/20767/how-useful-will-these-recommendations-be/120617 (2016-06-04)

[2] Expedia, Online travel booking. expedia.com (2016-06-04)

[3] Competition - Expedia Hotel Recommendations. https://www.kaggle.com/c/expedia-hotel-recommendations (2016-06-04)

[4] Mean Average Precision. https://www.kaggle.com/wiki/MeanAveragePrecision (2016-06-04)

[5] Wikipedia - Random forest https://en.wikipedia.org/wiki/Random_forest (2016-06-04)

[6] Machine learning for package users with R (5): Random Forest. http://tjo-en.hatenablog.com/entry/2015/06/04/190000 (2016-06-04)

[7] sklearn Random Forest Classifier documentation. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (2016-06-04)

[8] Optimizing your model with cross-validation, Kevin Markham. http://blog.kaggle.com/2015/06/29/scikit-learn-video-7-optimizing-your-model-with-cross-validation/comment-page-1/ (2016-06-04)

[9] Most Popular Local hotels script by Triskelion. https://www.kaggle.com/triskelion/expedia-hotel-recommendations/most-popular-local-hotels (2016-06-04)

[10] Clarification regarding the data leak by the competition admin. https://www.kaggle.com/c/expedia-hotel-recommendations/forums/t/20345/data-leak (2016-06-04)

VIII. Appendix

TABLE II: Expedia data fields for the training and test set

Column name	**Description**	**Data type**
date_time	Timestamp	string
site_name	ID of the Expedia point of sale (i.e. Expedia.com, Expedia.co.uk, Expedia.co.jp, ...)	int
posa_continent	ID of continent associated with site_name	int
user_location_country	The ID of the country the customer is located	int
user_location_region	The ID of the region the customer is located	int
user_location_city	The ID of the city the customer is located	int
orig_destination_distance	Physical distance between a hotel and a customer at the time of search. A null means the distance could not be calculated	double
user_id	ID of user	int
is_mobile	1 when a user connected from a mobile device, 0 otherwise	tinyint
is_package	1 if the click/booking was generated as a part of a package (i.e. combined with a flight), 0 otherwise	int
channel	ID of a marketing channel	int
srch_ci	Checkin date	string
srch_co	Checkout date	string
srch_adults_cnt	The number of adults specified in the hotel room	int
srch_children_cnt	The number of (extra occupancy) children specified in the hotel room	int
srch_rm_cnt	The number of hotel rooms specified in the search	int
srch_destination_id	ID of the destination where the hotel search was performed	int
srch_destination_type_id	Type of destination	int
hotel_continent	Hotel continent	int
hotel_country	Hotel country	int
hotel_market	Hotel market	int
is_booking	1 if a booking, 0 if a click	tinyint
cnt	Numer of similar events in the context of the same user session	bigint
hotel_cluster	ID of a hotel cluster	int

TABLE III: Expedia destination file

Column name	**Description**	**Data type**
srch_destination_id	ID of the destination where the hotel search was performed	int
d1-d149	latent description of search regions	double