Elias Ervelä, 518434 15.12.2022  
  
Git repository: <https://gitlab.utu.fi/emerve/machine-learning-project-course>

Machine Learning Project Course documentation

# Project objective

The projects objective is to guess the price of a house or an apartment, given all its features, using various machine learning models. We focused on Finnish apartment data. The project outline was to first find the data ourselves from internet or asking aparment dealers for the data. Then data cleaning/curation/imputation, model training, paramater tuning, and refining. This project was done as a course in Universtiy of Turku.

# Data acquisition

The data was web scraped from <https://asuntojen.hintatiedot.fi/haku/> thanks to Markus. This website has all the sold houses and apartments from the last 12 months in Finland. The website has the following features:

* Kaupunki
  + string
  + eg. “Turku”
* Huoneistotyyppi
  + string
  + “Yksiö”, “Kaksi huonetta”, “Kolme huonetta”, “Neljä huonetta tai enemmän”
* Kaupunginosa
  + string
  + eg. “Kohmo”
* Postinumero
  + string
  + eg. “20540”
* Huoneisto
  + string
  + eg. “3 h + k+ wc + k...”
  + every string is truncated
* Talotyyppi
  + string
  + “kt”, “ok”, “rt”
* m2
  + float
  + eg. 83.0
* Velaton hinta **(the feature we want to predict)**
  + float
  + eg. 128000
* €/m2
  + float
  + eg. 1542
  + direct proxy of predict value, so not used in prediction
* Rakennusvuosi
  + float
  + eg. 1992
* Kerros
  + string
  + eg. “2/4”
* Hissi
  + string
  + “on”, “ei”
* Kunto
  + string
  + “huono”, “tyyd.” or “hyvä”
* Tontti
  + string
  + “oma” or “vuokra”
* Energialuokka
  + string
  + D2013

# Data exploration

**Categorical features:**

Kaupunki   
Unique values: 178  
Missing values: 0

Huoneistotyyppi   
Unique values: 4  
Missing values: 0

Kaupunginosa   
Unique values: 2499  
Missing values: 2917

Postinumero   
Unique values: 852  
Missing values: 1857

Huoneisto   
Unique values: 10448  
Missing values: 52

Talotyyppi   
Unique values: 3  
Missing values: 0

Kunto   
Unique values: 4  
Missing values: 2982

Tontti   
Unique values: 3  
Missing values: 278

Energialuokka   
Unique values: 46  
Missing values: 5487

Postinumero and kaupunginosa have alot of missing values wich is pretty bad, since location is everyting when it comes to houses. Luckily they convey the same thing, so if one is missing, the other one can be deducted. Kunto and energialuokka have also alot of missing values wich is a bad thing, since those things might also affect the house price quite alot.

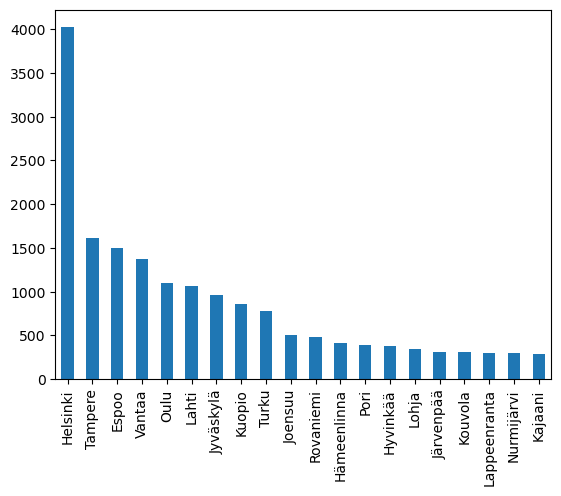
**Numeric features:**

m2  
Missing values: 0

Velaton hinta  
Missing values: 0

€/m2  
Missing values: 0

Rakennusvuosi:   
Missing values: 0

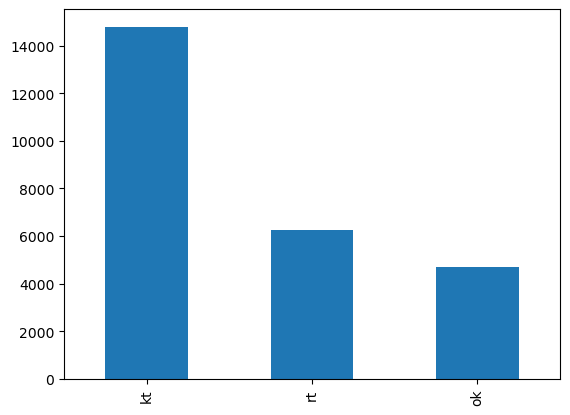
**# Amount of data per city, biggest 20**  
Helsinki 4023  
Tampere 1617  
Espoo 1498  
Vantaa 1378  
Oulu 1098  
Lahti 1060  
Jyväskylä 957  
Kuopio 857  
Turku 776  
Joensuu 503  
Rovaniemi 485  
Hämeenlinna 408  
Pori 387  
Hyvinkää 376  
Lohja 349  
Järvenpää 315  
Kouvola 312  
Lappeenranta 304  
Nurmijärvi 297  
Kajaani 293  
  
The largest cities are really large. The imbalance in the data might be a problem.

Kaksi huonetta 8228  
Neljä huonetta tai enemmän 7637  
Kolme huonetta 6820  
Yksiö 3065

Chart, bar chart

Description automatically generated

This is balanced enoguh, I’d say. Altough the “Neljä huonetta tai enemmän” hides alot of other values behind it.



Omakotitalot might be hard case, since theres alot less data and much more variation in these cases.

# Data pre-processing/curation

**Kaupunki**If kaupunki would be one-hot encoded, there would be 178 new columns. And there would be a data problem since there are multiple cities with very little data. There are 24 cities with 5 or less datapoints, 44 cities with 10 or less datapoints. So treating this as a categorical might not be robust to small cities. Thats why I turned the city column into stats of the city. From tilastokeskus (<https://pxdata.stat.fi/PXWeb/pxweb/fi/Postinumeroalueittainen_avoin_tieto/Postinumeroalueittainen_avoin_tieto__2022/paavo_pxt_12f8.px/>), I harvested population density (Asukkaat yhteensä, 2020 (HE) / Kunta pinta-ala) and density of service jobs (Palveluiden työpaikat, 2019 (TP) / Kunta pinta-ala). I figured these would be the most telling signs of citys. If we want to improve results, we could try to find more features from there.

**Postinumero**There were a lot of missing postal codes, BUT there were “Kaupunginosa” present in some of those. I had a database of finnish postal codes and their respective names. Rows where I found a unique match in city and in postal code name, I added the respective postal code in the row. With this I was able to fix 655 rows. Then there were some cases where it got multiple matches like from “Parola” I got Parola and Parolannummi, and in these cases I took the postal code of the smaller name. Some got multiple matches from names thad had “Asemanseutu” version of an area so I these I added to the non-Asemaseutu. The rest (382) where quite ambiquios, so these Postal Codes I left as nan.

These postal codes I turned into:  
- Asuintiheys (Asukkaat yhteensä, 2020 (HE) / Postinumeroalueen pinta-ala)  
- Asukkaiden mediaanitulot  
- Palveluiden työpaikat tiheys (Palveluiden työpaikat, 2019 (TP) / Postinumeroalueen pinta-ala)  
- Työttömien osuus (Työttömät, 2019 (PT) / Työttömät + Työlliset)  
- (Koulutus (Koulutetut yhteensä, 2020 (KO) / Asukkaat yhteensä, 2020 (HE)))

**Huoneiden lukumäärä**Where huoneistotyyppi was yksiö -> 1, kaksio -> 2, kolmio ->3.  
And then the rest were 4 tai enemmän. These I tried to parse from the “Huoneisto” column. And the tactic was a regex army. What made it extra hard was that the string value in that column was cut off. The regex rows I had is too many to paste here. See git for those.

Removed rows that were car places or other similar I found.  
- those which start with "auto" (auto.\*$)  
- including "katos", but not including "+" or "," (^.\*katos.\*$, not ^.\*[+,].\*$)  
-one "pihapaikka"

- Values like 3-4h, I took the larger number, since most of 3-4 were marked as "Neliö tai enemmän" rather than "Kolme huonetta"  
- And those that had living room marked seperately I summed +1  
- eg. ^.\*(3|3h) \*- \*(4|4h).\*$ AND ^.\*oh.\*$ -> 5 rooms  
- eg. ^.\*(3|3h) \*- \*(4|4h).\*$ AND NOT ^.\*oh.\*$ -> 4 rooms  
- there were also couple special cases I found like "4(-5)h", "4-7h" where I took the larger

Then the rest was done like:  
- eg. ^.\*5 \*h.\*$ AND ^.\*oh.\*$ AND the room wasn't specified with the rules above -> 6 rooms  
- same, but with 5mh -> 6 rooms  
- eg. ^.\*5 \*h.\*$ AND NOT ^.\*oh.\*$ AND the room wasn't specified with the rules above -> 6 rooms  
-without h or mh, but with "," (4,.\*$) -> 4 rooms  
-without h or mh, but with "+" (4\+.\*$) -> -> 4 rooms

The amount of rows that I coudn't determine the n. of rooms: 196  
These included like:  
k, oh, 2 x mh, ...  
2 x eteisaula +...  
4k,k,p  
6 rivitalohuone...  
4 MH + Kirjasto...  
4  
Oh, k, mh, 2xas...  
4mh+työh.+...  
4 mh,työh....  
26 asuntoa  
2 erillistaloa,...  
6r k oleskeluti...  
Muissa tiloissa...  
tupakeittiö...  
Tupa,k,mh,parvi...  
Keittiö,Sa...

**Sauna**:  
To binary. Has sauna -> 1, otherwise ->0.  
This was tackled with more of a regex platoon.  
Has space, + or , before and after s somewhere  
str.match("^.\*[ ,+/(-.]s[ ,+/)-.].\*$", case=False)] = 1  
str.match("^.\*[ ,+/(-.]sa[ ,+/)-.].\*$", case=False)] = 1  
str.match("^.\*[ ,+/(-.]sau[ ,+/)-.].\*$", case=False)] = 1  
str.match("^.\*[ ,+/(-.]saun[ ,+/)-.].\*$", case=False)] = 1  
str.match("^.\*[ ,+/(-.]s$", case=False)] = 1  
str.match("^.\*kh-s.\*$", case=False)] = 1  
str.match("^.\*kph-s.\*$", case=False)] = 1

**Talotyyppi:**This had 3 unique values. I decided to one-hot encode these due to the low amount of unique values.

**Rakennusvuosi:**Removed all houses older than 1800. There were a few of these

**Kerros:**This was a string like “5/7”. I parsed the first and the last number and turned these into two columns “Kerros” and “Kerros\_max”.  
Values that have greater value in “Kerros” -> nan.  
Where kerros is nan and the house type is ‘ok’ or ‘rt’ -> 1.

**Tontti:**Turned this to binary: oma -> 1, vuokra -> 0. And nan -> 1, since its more common in all the cities.

**Energialuokka:**  
Turned this in to numerical. A->1, B->2, ..., G->7. This had a lot of nan values. I didn’t have a great plan for that.

**m2**  
Houses that had less than 10 m^2, I removed.

**Missing data**Out of 8 813 of 25 750 rows had missing values. At first I removed all these rows. But in the later stages stacked model phase, I used an kNN imputer to impute the missing values.

After data cleaning and curation, these were the 19 features I used.  
'm2', 'Rakennusvuosi', 'Kunto', 'Energialuokka', 'PN\_Asuintiheys', 'PN\_Palvelutyötiheys', 'PN\_Työttömien osuus', 'PN\_Mediaanitulot', 'KA\_Asuintiheys', 'KA\_Palvelutyötiheys', 'Kerros', 'Kerros\_max', 'Tontti', 'Hissi', 'kt', 'ok', 'rt', 'Huoneiden\_lkm', 'Sauna'.  
And the value we try to predict is the “Velaton hinta”.  
Number of rows: 18 024

# First model architecture, first results

I had nested cross validation scheme to determine the best model without data leakage. I used only data without missing values in these runs.

First I tried RandomForestRegressor.  
I did 12 fold nested cross validation with these models:

max\_features = 1.0 or "sqrt"  
bootstrap = True or False  
min\_samples\_split = 2 or 6

RandomForestRegressor(n\_estimators=400, max\_features=1.0, bootstrap=True, min\_samples\_split=2),  
RandomForestRegressor(n\_estimators=400, max\_features=1.0, bootstrap=False, min\_samples\_split=2),  
RandomForestRegressor(n\_estimators=400, max\_features="sqrt", bootstrap=True, min\_samples\_split=2),  
RandomForestRegressor(n\_estimators=400, max\_features="sqrt", bootstrap=False, min\_samples\_split=2),  
RandomForestRegressor(n\_estimators=400, max\_features=1.0, bootstrap=True, min\_samples\_split=6),  
RandomForestRegressor(n\_estimators=400, max\_features=1.0, bootstrap=False, min\_samples\_split=6),  
RandomForestRegressor(n\_estimators=400, max\_features="sqrt", bootstrap=True, min\_samples\_split=6),  
RandomForestRegressor(n\_estimators=400, max\_features="sqrt", bootstrap=False, min\_samples\_split=6).

The most chosen model was:

RandomForestRegressor(n\_estimators=400, max\_features="sqrt", bootstrap=False, min\_samples\_split=2),

with 11/12 inner iteration wins.  
*Avg test R^2 score*: 0.9045644963139403   
(all iterations: [0.92321672 0.91553817 0.84092942 0.90443517 0.92704918 0.84626635 0.93475144 0.91428546 0.9124305 0.90014655 0.93116051])  
*Average 15% margin accuracy*: 0.6747767723377479 (prediction is within 15% of the real price)  
(all iterations: [0.65985498 0.64469347 0.67831246 0.6723797 0.69676994 0.6723797 0.69940672 0.66842452 0.67831246 0.66183256 0.69017798])

This model had feature importances of:  
'm2', 0.20381658  
'Rakennusvuosi', 0.09507726  
'Kunto', 0.01294798  
'Energialuokka', 0.02165763  
'PN\_Asuintiheys', 0.07489879  
'PN\_Palvelutyötiheys', 0.06730169  
'PN\_Työttömien osuus', 0.06089951  
'PN\_Mediaanitulot', 0.10630057  
'KA\_Asuintiheys', 0.09718269  
'KA\_Palvelutyötiheys', 0.08195093  
'Kerros', 0.01591861  
'Kerros\_max', 0.01808567  
'Tontti', 0.00417479  
'Hissi', 0.00571097  
'kt', 0.0090097   
'ok', 0.01014729  
'rt', 0.00429126  
'Huoneiden\_lkm', 0.09940017  
'Sauna' 0.01122789

The R^2 value is suprisingly good. But more suprisingly the 15% margin score is quite low. The feature importances tell that the size affects the most, but the location data has also a great deal affecting.

# Second model results

# The second model I quickly tried was support vector machine regressor. I tried with three different kernels, rbf, poly, sigmoid.

And with a quick cross validation I got R^2 value of 0.8503976757957813 with rbf kernel. So not as good as random forest.

# Thrid model results

I tried XGBoost and did nested corss validation on couple of different parameters:

models = [

XGBRegressor(n\_estimators=400, max\_depth=7),

XGBRegressor(n\_estimators=400, max\_depth=0),

XGBRegressor(n\_estimators=400, max\_depth=7, colsample\_bytree =0.6),

XGBRegressor(n\_estimators=400, max\_depth=0 colsample\_bytree =0.6),

].

After nested cross-validation The best scoring one was XGBRegressor(n\_estimators=400, max\_depth=7), with score 0.9015057384694751.

Basically the same score compared to the Random Forest Regressor.

# RandomForestRegressor on stratified data

I tried the best performing model on different subsets of data by cross-validation. The scores were the following.

All data 0.9045644963139403

Only Helsinki 0.8839437449662586  
Only Turku 0.8236563146382201  
Only Tampere 0.8657461046903183  
Ten biggest cities 0.9076997237265656  
20 biggest cities 0.9132089082868516

I would have guessed the only Helsinki/Tampere/Turku would have performed better. But it seems the amount of data really has an effect.

Only Omakotitalo 0.7642279040350036  
Only Kerrostalo 0.9302072126906881  
Only Rivitalo 0.8835765076506332

This result is quite expected. There’s less data on omakotitalot and much more variation on different kinds of houses, and the limited features might not convey the things that affect the price. The most amount of data is on kerrostalot, so this is the reason it’s score is the best.

# Boosting the model

With random forest regressor and AdaBoost with 10 estimators, I was able to achieve 0.9037213950859929 score, which is not any better than the plain random forest regressor. I'm thinking this is the ceiling with the data, and we can't get better than this with the data. There’s many factors that affect the house price thats present not in our dataset.

# Stacked model

We tried a stacked model with multiple models. I faced in to many techincal difficulties with this. I had done my preprocessings in Pandas arrays and had to turn them in to Numpy operations, and my original preprocessings was leaning in to dropping rows, and the stacked model wasn’t quite compatible with this. I had to have an imputer in the end.  
In the end I was able to score of 0.8931602355046816 with all the working models of our group. Note that it is worse score than my since I had drop rows that had missing values, so this score includes rows that had missing values imputed. My model alone in this had a score of 0.84.

estimators = [

('Elias\_RFR', Elias\_RandomForestRegressor),

#('Elias\_XGB', Elias\_xgbr), **<- couldn’t get this to work, not a sklearn model so not compatible**

('JR\_CatBoost', JR\_CatBoost),

('JR\_GB', JR\_GB),

('MR\_GB', MR\_GB),

('MR\_MP', MR\_MP),

("MM\_RF", MM\_RF),

("MM\_GB", MM\_GB),

#("SP\_wrapped\_ada", SP\_wrapped\_ada), **<- couldn’t get this to work, due to older version**

#("SP\_wrapped\_extratrees", SP\_wrapped\_extratrees), **<- couldn’t get this to work, due to older version**

("AV\_GB", AV\_GB),

("AV\_RF", AV\_RF),

]

0.8931602355046816

# Conclusions

I’d say we accomplished the best we could with this dataset. After trying many models with many parameters, and after trying boosting methods and stacking methods, I’d say, the we’ve reached very close to the best possible score with this dataset. Many things affect the price of the house, that we just don’t have knowledge of in this dataset, for example “vastike”, the full “huoneisto” text, more spesific location, renovations and more. The model we achieved can give a great ballpark guess of the price area, but not determine the final price without a human seller and human buyer. This could be used as online tool for people thinking of selling or buying an house, and giving them intuition on house price. And maybe give info about the model in advance, that this is more accurate on bigger cities and with flats.