Roti, Kapda aur Makan

Beginner's Machine Learning Competition on Cerebro

Name: Rahul Sheshanarayana (username: 2pac)

Branch: Polymer Science and Engineering (2nd yr)

Phone No.: +91 6362870308

Email: rsheshanarayana@ee.iitr.ac.in

Github: https://github.com/RSIITR/RKM-Cerebro

Competition Description

"Roti Kapda aur Makaan" (Bread, cloth and house), everyone dreams of buying his/her own house and one of the most important factors while buying any piece of property is the price of it according to its location, type, etc. Your task, if you choose to accept, is to predict the price of houses on the basis of several factors provided in the dataset. The training dataset contains 13 self-explanatory features such as the location of the property, property type, Property Age (old vs new) etc. Using this data, you need to make predictions for the price of a particular property on the test set.

Data Description

Following are the explanation of the various columns present in the dataset of Roti, kapda aur makaan challenge: -

TID - ID of the instance

Date - Date on which the data about a particular property was recorded

Property Type - categorical description of the type of property such as flat, land etc

OldvNew - tells us if the property is old or new

Duration - categorical description of the duration of the ownership of the property

AddressLine1 - address line 1 of the property

AddressLine2 - address line 2 of the property

Street - street on which property is located

Locality - locality of the property

Town - town of the property

Taluka - taluka (administrative district for taxation purposes) of the property

District - district of the property

Postcode - postal code of the property

Price - Target variable i.e. price of the property

Price Category - price category

Imports and Libraries used

Pandas

Numpy

Matplotlib.pyplot as plt

XGBRegressor from xgb

LGBMRegressor from lightgbm

Loading and Pre-processing the data

Loaded the train and test data using pandas .csv reader (pd.read_csv()) and analysed the different columns.

Checked for null values using .**isnull().sum()** function to find the columns which can consist of a lot of missing values. Columns: **AddressLine2 and Locality** were dropped due to NaN values being more than 50% in number. TID was however dropped as ID columns have no trend to show during training.

The other columns with less null values were filled in with their mode values using **.fillna().**

Feature Engineering

Feature engineering was kept simple, by encoding the categorical variables.

The date column was dropped after splitting it into day, month and year columns using .**str.split()**.

One hot encoding was used to encode columns: **PropertyType, OldvNew, Duration and PriceCategory**.

Frequency encoding was used to encode much vague columns**: AddressLine1, Taluka, Town, District, Street, PostCode.**

Model selection and prediction

Data Visualization was done with scatter plots, which was seen to be linear w.r.t label (y).

Since the data is relatively big, tree-based models would work the best here, with training time kept in mind. Linear models like **Linear Regression and SVM** would have performed well, but would have proven to be costly by taking a lot of time to fit this data.

After trying all possible models, the best one turns out to be a **weighed ensemble** of **lighgbm** and **xgboost** models, with 50% weightage for each.

The only hyperparameter triggered in the 2 models was **n_estimator**, the others were in their default values.

Submission

After getting the prediction from the above model, a dataframe was created with **TID** from test data and the above predictions as **Price**.

*I tried out the above submission for 16 times with some changes and ended up with a very bad score, until I realized my mistake of not looking into the sample submission file, the data was arranged in ascending order by **TID.**

Used **.sort_values(by = 'TID', ascending = 1)** to group the data in ascending order by **TID** and used **.reset_index()** to reset the index from 0.

Finally submitted the above dataframe by converting it into .csv using **.to_csv()** and setting index as False. Scored a score of mse = **11500** on public data.

Stayed at the top until I became over ambitious to score better. Tweaked (increased a lot) **n_estimators** of xgboost model, ended up scoring 22000 ☹ (overfitting).

Tried to tweak the xgboost model again by decreasing the increased n_estimators and increased that of lightgbm model, **Boom! Mse = 11000.** Left the score there without being **overambitious.**