Work Integrated Learning Programmes | BITS Pilani — Pilani | Dubai | Goa | Hyderabad

Computer Science & Information Systems
Machine Learning – Lab Report 1

Name: Avadhesh Singh **ID:** 2020hs11508@wilp.bits-pilani.ac.in

Module 1: End to end ML Project

1. Objective

- The objective is to perform end to end Machine Learning project on a dataset.

2. Pre-requisite

- **Tool: -** Python3, Jupyter Notebook
- **Libraries required: -** numpy, pandas, matplotlib, sklearn
- **Input: -** the California Housing Prices dataset from the StatLibrepository
- **ML Models: -** Linear Regression, Decision Trees, Random Forests, Support Vector Machines

3. Steps performed

Steps 1: Get the data.

- Downloaded the data from URL: https://raw.githubusercontent.com/ageron/handson-ml2/master/datasets/housing/housing.tgz.
- Explored the data structure
  - There are 20,640 instances in the dataset.
  - total_bed rooms attribute has only 20,433 non-null values, so 207 districts are missing this feature.
  - All attributes are numerical, except the ocean_proximity field.
  - The values in that column were repetitive, which means that it is probably a categorical attribute.
  - The value_counts() method for what categories and how many districts belong to each category.
  - The describe() method shows a summary of the numerical attributes
- Splitted the dataset in to Training and Test data Set

Steps 2: Data Analysis and visualizing the data to gain insights.

- Visualizing geographical data
  - Since there is geographical information (latitude and longitude), creating a scatterplot of all districts to visualize the data
  - From the plot we have observed the high-density areas & high price areas are the Bay Area.
- Looking for correlations
  - Computed the standard correlation coefficient (also called Pearson's r) between every pair of attributes using the corr() method.
  - Observed that the median house value tends to go up when the median income goes up. While the latitude values is having negative correlation.
  - For more understanding used scatter_matrix function, which plots every numerical attribute against every other numerical attribute.
- After Experimenting with Attribute combinations and created new attributes. (rooms_per_household, bedrooms_per_room, population_per_household)

    - Because the total number of rooms in a district is not very useful. Instead, rooms per household is more important in housing price. Similarly other 2 feature drawn.

Steps 3: Data Wrangling: Prepare the data for Machine Learning algorithms.

- Data Cleaning: Handling missing features,
    - Get rid of the corresponding districts.
    - Get rid of the whole attribute.
    - Set the values to some value (zero, the mean, the median, etc.). We can accomplish these easily using DataFrame's dropna(), drop(), and fillna() methods:
- For Handling text and categorical attributes two methods experimented.
    - **Method 1:** Changing categorical values to numerical values using OrdinalEncoder
    - This will provide numerical values based on number of categories available.
    - This may not be helpful in case one category has similar feature as other.
    - **Method 2:** Using OneHotEncoder, generally it returns sparse matrix
- Custom Transformers is used to add extra combined attributes.
- As numerical attribute has different-different scales. So, Feature scaling is performed to set all attribute to same scale. (Min-max scaling or Standardization). In this case Standardization is performed.
- Transformation Pipelines: A list of transformers is provided, and when its transform() method is called it runs each transformer's transform() method in parallel, waits for their output, and then concatenates them and returns the result (and calling its fit() method calls all each transformer's fit() method). A full pipeline used for handling both numerical and categorical attributes.

Steps 4: Selecting and Training a Model

- **LinearRegression:** Measuring this regression model's RMSE on the whole training set using Scikit-Learn's mean_squared_error function:
  RMSE: - 68628.19819848922
- **DecisionTreeRegressor:** Training a DecisionTreeRegressor. This is a powerful model, capable of finding complex nonlinear relationships in the data
  RMSE: - 0.0; Here Error is zero. It seems, it is much more likely that the model has badly overfit the data.
- Better Evaluation Using Cross-Validation
  K-fold cross-validation: it randomly splits the training set into 10 distinct subsets called folds, then it trains and evaluates the Decision Tree model 10 times, picking a different fold for evaluation every time and training on the other 9 folds. The result is an array containing the 10 evaluation scores:
  After cross-validation DecisionTreeRegressor RMSE: 71407.68766037929
  After cross-validation LinearRegression RMSE: 69052.46136345083
  Now the Decision Tree doesn't look as good as it did earlier. It seems to perform worse than the Linear Regression model.
- **RandomForestRegressor:**
  The RandomForestRegressor. Random Forests work by training many Decision Trees on random subsets of the features, then averaging out their predictions.
  RMSE: 18603.515021376355
  After cross-validation RMSE: 50182.303100336096
- **SVM:** RMSE: 111094.6308539982

Steps 5: Fine-tune of the Model

- Grid Search is used to find great combination of hyperparameter values. Here, It has explored 12+6=18 combination of hyperparameter values. For 5 fold CV, total 18*5 round of training.
- Randomized Search CV is used when hyperparameter space is large.
- Analyze the Best Models and Their Errors: used for relative importance for each attribute for making accurate prediction.
- Evaluate Your System on the Test Set
  - Now is the time to evaluate the final model on the test set.
  - We get the predictors and the labels from our test set, run our full_pipeline to transform the data
  - Call transform() and evaluate the final model on the test set

4. Results

- **Final Model selected: -** RandomForestRegressor

On Test Data:

- **Final Model RMSE: -** 47730.22690385927
- **95% confidence interval: -**
  - T-score: [45685.10470776, 49691.25001878]
  - Z-score: [45685.717918136455, 49690.68623889413]

5. Observation & Conclusion

- The data was thoroughly understood and first Data Analysis and visualization has been performed.
- After that to gain more insight, some new attribute has been added.
- After data wrangling performed. Missing values handled using median, Categorical values changed to numerical values, and standardization done.
- Used different ML algorithms **LinearRegression, DecisionTreeRegressor, RandomForestRegressor and SVM** for training**.** Out of which **RandomForestRegressor** is selected as final model due to less RMSE values than others in validation.
- This ML model trained and fine-tuned.
- Finally tested on Test dataset.