Machine Learning – Training Models

Michael Claudius, Associate Professor, Roskilde
Jens Peter Andersen, Assistant Professor, Roskilde

10.02.2021, revised 23.02.2022

Machine Learning Project

- ***Machine Learning has a number of phases***
- ***The phases can be overlapping and/or iterative***

1. **Look at the big picture.**
2. **Get the data.**
3. **Discover and visualize the data to gain insights.**
4. **Prepare the data for Machine Learning algorithms.**
5. **Select a model and train it.**
6. **Fine-tune your model.**
7. **Present your solution.**
8. **Launch, monitor, and maintain your system.**

- **A detailed checklist is given on ML Management Checklist (PDF)**

- **Remember always adapt the order and the checklist to your needs**

The Context: Housing prices

- **California median housing price for a block group**
- **Block group: unit population of 600-3.000 people**
- **Data size app. 20.000**

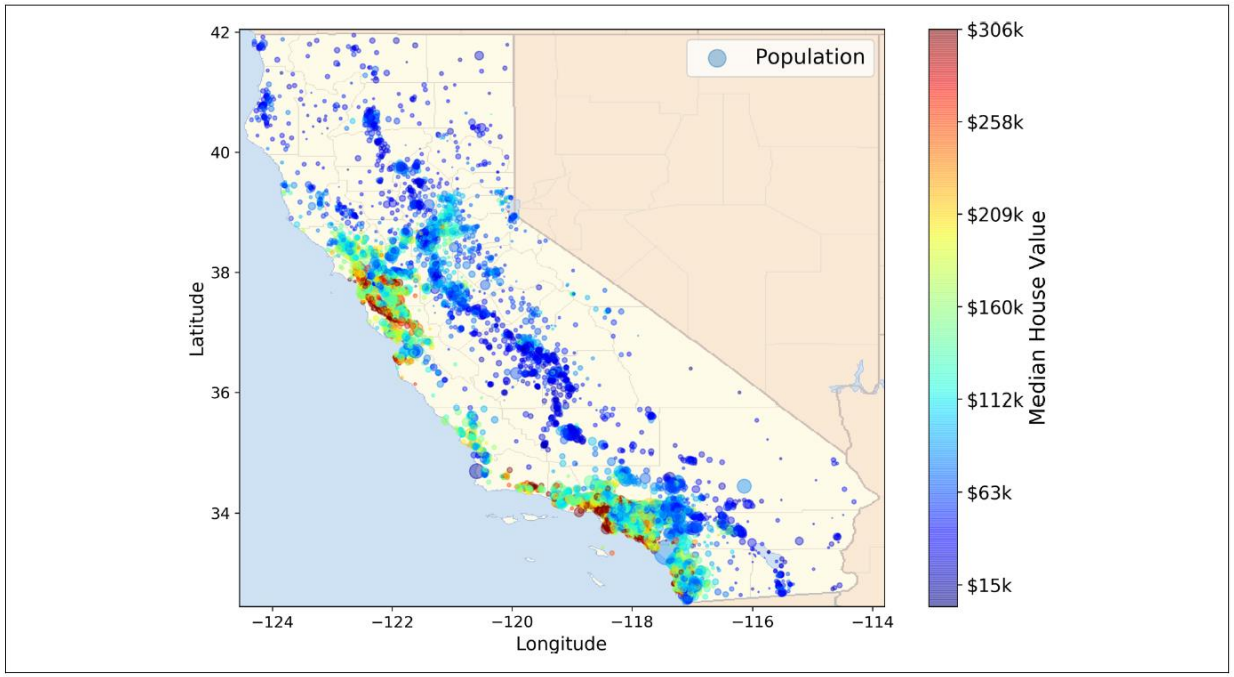


Figure 2-1. California housing prices

Machine Learning Prepare Data

Its about making data ready to be used by Machine Learning algorithms

1. **Data Cleaning**: Handle missing feature instances -> How to fix it
2. **Handle Text and Categorical Attributes**: Convert values to numbers so they can handled by ML algorithms
3. **Feature Scaling**: Scale features to the same order of magnitude: E,g intervals -1..1 or 0..1 or distribution around median
4. **Custom Transformers**: Custom transformer classes on data can be defined to be used in e.g. transformations pipelines
5. **Transformation Pipelines**: Feature from Scikit-Learn, that automatically e.g. can apply the above mentioned transformers

Machine Learning Select and Train Model

Its about finding the best model to launch.

1. Select and Train a Model on Training Set
- Training and Evaluating on the Training Set
- Better Evaluation Using Cross-Validation

2. Fine-Tune Your Model
- Grid Search
- Randomized Search
- Ensemble Methods

3. Analyze the Best Models and Their Errors
- Importance of attributes for predictions
- Evaluate Your System on the Test Set

4. Launch, Monitor, and Maintain Your System
- In real world a complex and important point !

Select and Train a Model

- **Training and Evaluating on the Training Set**: Do the actual training of the model.
- **Better Evaluation Using Cross-Validation**: Exploit the feature set thoroughly.

Training the Training Set

Problem: How to predict mean house values. Which model to train/choose first. Keep it simple ☺
Solution: Apply a linear regression model provided by Scikit-Learn
Check up: Look at a small data set to see predictions
:

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(housing_prepared, housing_labels)

some_data = housing.iloc[:5]
some_labels = housing_labels.iloc[:5]
some_data_prepared = full_pipeline.transform(some_data)

print("Predictions:", lin_reg.predict(some_data_prepared))
Predictions: [ 210644.6045  317768.8069  210956.4333  59218.9888  189747.5584]
print("Labels:", list(some_labels))
Labels: [286600.0, 340600.0, 196900.0, 46300.0, 254500.0]
```

Evaluating on the Training Set

Problem: Select a performance measurement (using the distance between predicted and actual values)
Solution: Apply the Root Mean Square Error formula (RMSE)

Steps in Python – e.g.:

```
from sklearn.metrics import mean_squared_error
housing_predictions = lin_reg.predict(housing_prepared)
lin_mse = mean_squared_error(housing_labels, housing_predictions)
lin_rmse = np.sqrt(lin_mse)
lin_rmse = 68628.19819848922
```

Evaluate: Very bad! `lin_rmse = 68628`

Solution: Try another model: Apply Decision Tree Regressor

```
tree_rmse = 0.0
```

Evaluate: Too good to be true! Overfitting
Solution: Use k-fold cross evaluation

Better Evaluation Using Cross-Validation

Idea: Utilizing the feature set for training and validation

Solution: Scikit-Learn's K-fold cross-validation feature could be used:
Splits the training set into e.g.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.

Performance: `lin_rmse = 68628 tree_rmse = 71407`
Evaluate: Very bad!

Solution: Try another model: Apply Random Forest Regressor
Performance: `forest_rmse = 50182`
Evaluate: OK, let's fine tune this model

Fine-Tune Your Model - Methods

Purpose: How to make a promising learning model perform better? (e.g. a Random Forest Regressor model)

Random Forest: Using the same training algorithm (decision tree) for every predictor
Built on a number of decision trees
Data are picked randomly in samples used on each tree (the predictor)
Training is done on different random subsets (i.e. the samples)
Finally chooses the maximum voting (classification) or averages of predictions (regression)

Solution: Variations of hyper parameters (`n_estimators`, `max_features`, `bootstrapping = false/true`)
n_estmators: Number of decision trees to be built before taking the maximum voting or averages of predictions
bootstrapping: *true* allows training instances to be sampled several times for the same predictor.

Methods:

Gridssearch: Specify few combinations of hyper parameters and values.
Random search: Try out many combination (1000 iterations). Really slow !
Ensemble Methods: Combine several models, Advanced!

Fine-Tune Your Model – Grid Search

Steps in Python could be like this:

```
from sklearn.model_selection import GridSearchCV

param_grid = [
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
  ]

forest_reg = RandomForestRegressor()

grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)

grid_search.fit(housing_prepared, housing_labels)
```

Grids Search – Hyper Parameters

- **Find the optimal hyper parameters:**

```
>>> grid_search.best_params_
{'max_features': 8, 'n_estimators': 30}

>>> cvres = grid_search.cv_results_
>>> for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
...     print(np.sqrt(-mean_score), params)
...
63669.05791727153 {'max_features': 2, 'n_estimators': 3}
55627.16171305252 {'max_features': 2, 'n_estimators': 10}
53384.57867637289 {'max_features': 2, 'n_estimators': 30}
60965.99185930139 {'max_features': 4, 'n_estimators': 3}
52740.98248528835 {'max_features': 4, 'n_estimators': 10}
50377.344409590376 {'max_features': 4, 'n_estimators': 30}
58663.84733372485 {'max_features': 6, 'n_estimators': 3}
52006.15355973719 {'max_features': 6, 'n_estimators': 10}
50146.465964159885 {'max_features': 6, 'n_estimators': 30}
57869.25504027614 {'max_features': 8, 'n_estimators': 3}
51711.09443660957 {'max_features': 8, 'n_estimators': 10}
49682.25345942335 {'max_features': 8, 'n_estimators': 30}
62895.088889905004 {'bootstrap': False, 'max_features': 2, 'n_estimators': 3}
54658.14484390074 {'bootstrap': False, 'max_features': 2, 'n_estimators': 10}
59470.399594730654 {'bootstrap': False, 'max_features': 3, 'n_estimators': 3}
52725.01091081235 {'bootstrap': False, 'max_features': 3, 'n_estimators': 10}
57490.612956065226 {'bootstrap': False, 'max_features': 4, 'n_estimators': 3}
51009.51445842374 {'bootstrap': False, 'max_features': 4, 'n_estimators': 10}
```

- **Final result:** max_features = 8 and n_estimators = 30

Analyze the best model

Analyze the Importance of attributes for predictions

```
>>> sorted(zip(feature_importances, attributes), reverse=True)
[(0.3661589806181342, 'median_income'),
 (0.1647809935615905, 'INLAND'),
 (0.10879295677551573, 'pop_per_hhold'),
 (0.07334423551601242, 'longitude'),
 (0.0629090704826203, 'latitude'),
 (0.05641917918195401, 'rooms_per_hhold'),
 (0.05335107734767581, 'bedrooms_per_room'),
 (0.041143798478729635, 'housing_median_age'),
 (0.014874280890402767, 'population'),
 (0.014672685420543237, 'total_rooms'),
 (0.014257599323407807, 'households'),
 (0.014106483453584102, 'total_bedrooms'),
 (0.010311488326303787, '<1H OCEAN'),
 (0.002856474637320158, 'NEAR OCEAN'),
 (0.00196041559947807, 'NEAR BAY'),
 (6.028038672736599e-05, 'ISLAND')]
```

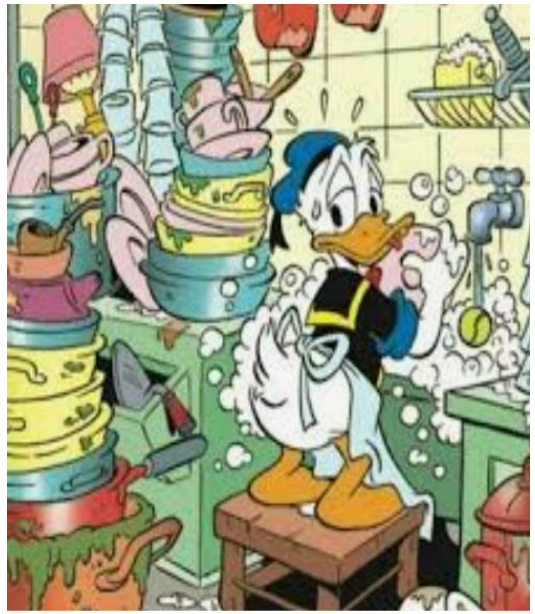
Cut some attributes: E.g. 2% or 5% level

Evaluate Your System on the Test Set

Exercise

- **It is time for discussion and training models.**
- **We will again investigate the housing project !!**

  - Linear Regression Standard
    Linear Regression Missing Data
  - Housing Ch. 2 No. 2



- ***Look at hyper parameters, but don't loose the overview ☺***
- ***It is a hard job to train and find the best model.***