# Model selection and evaluation

Alpha Hamadou Ibrahim

# Outline

- Introduction
- Exploratory data analysis
- Machine learning
  - Model selection
  - Parameters tuning
  - Model evaluation
- Questions

# Introduction

Given  a set of variable,

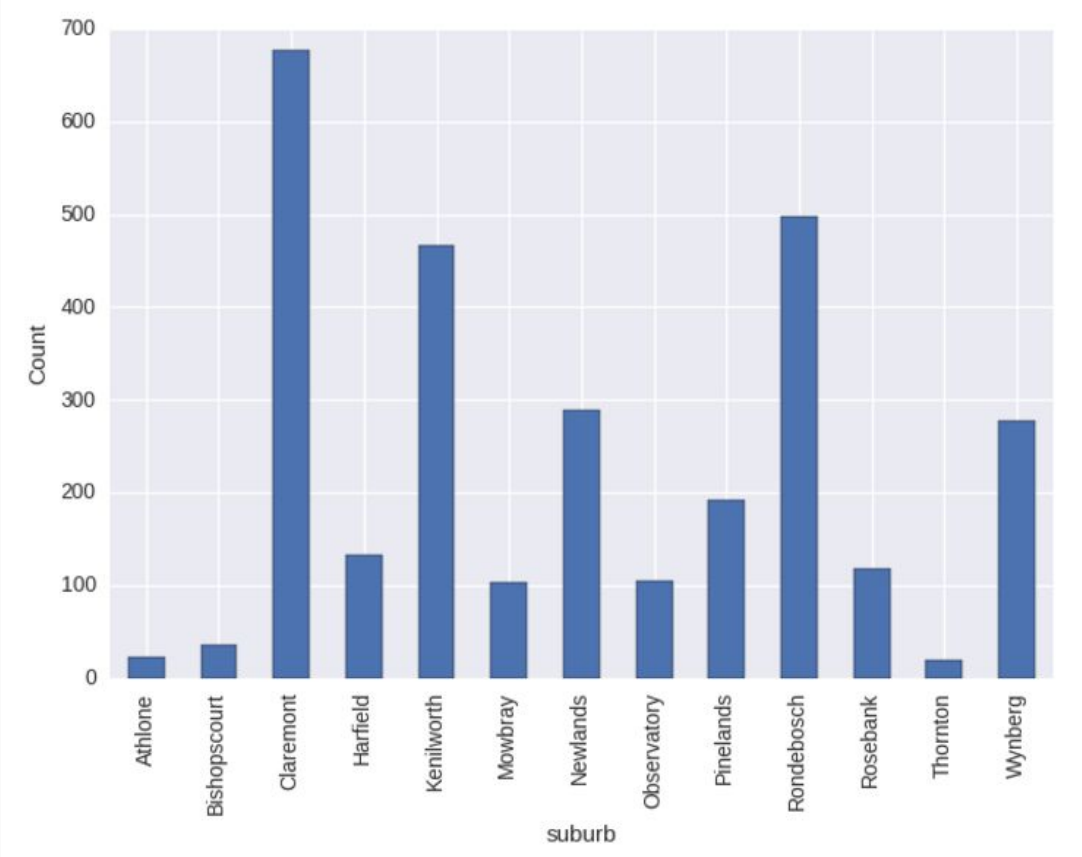
how accurately can one predict the value of a property?

# Exploratory data analysis

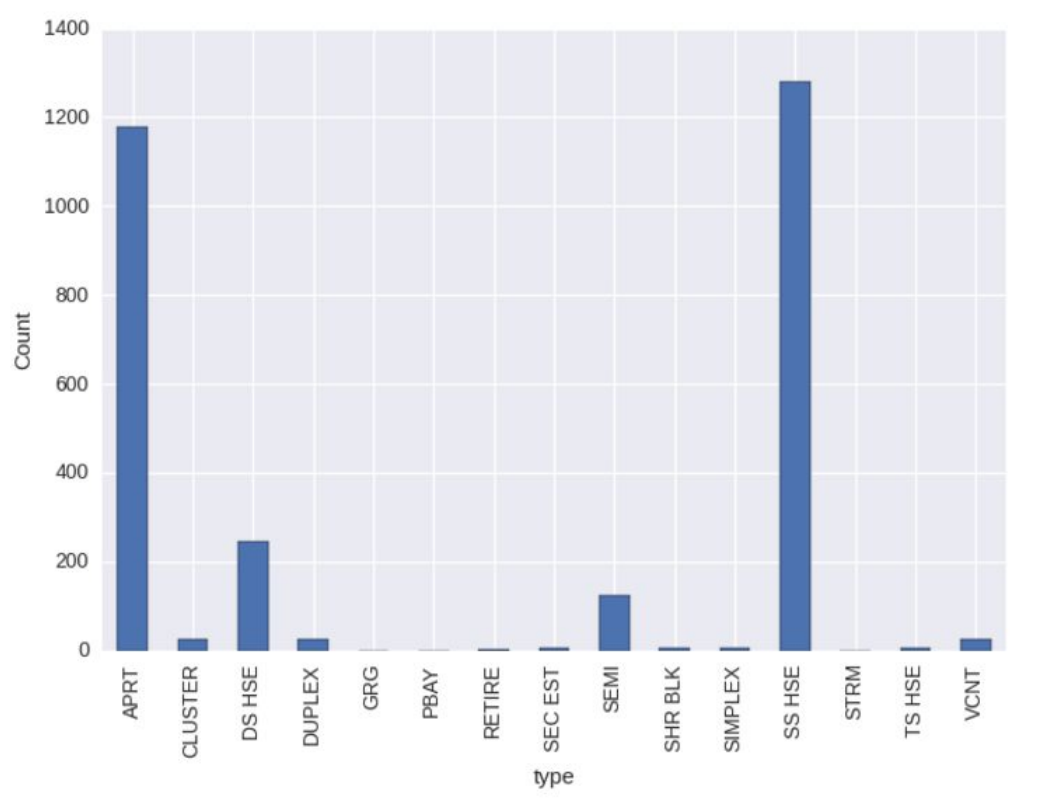
The data contains

- About 2939 properties
- Property details
  - Suburbs, type, land size, number of rooms, etc
- Other details
  - Sale details (source of information, listed price, sale price, finance type, etc)
  - Buyer details (eg Nationality)

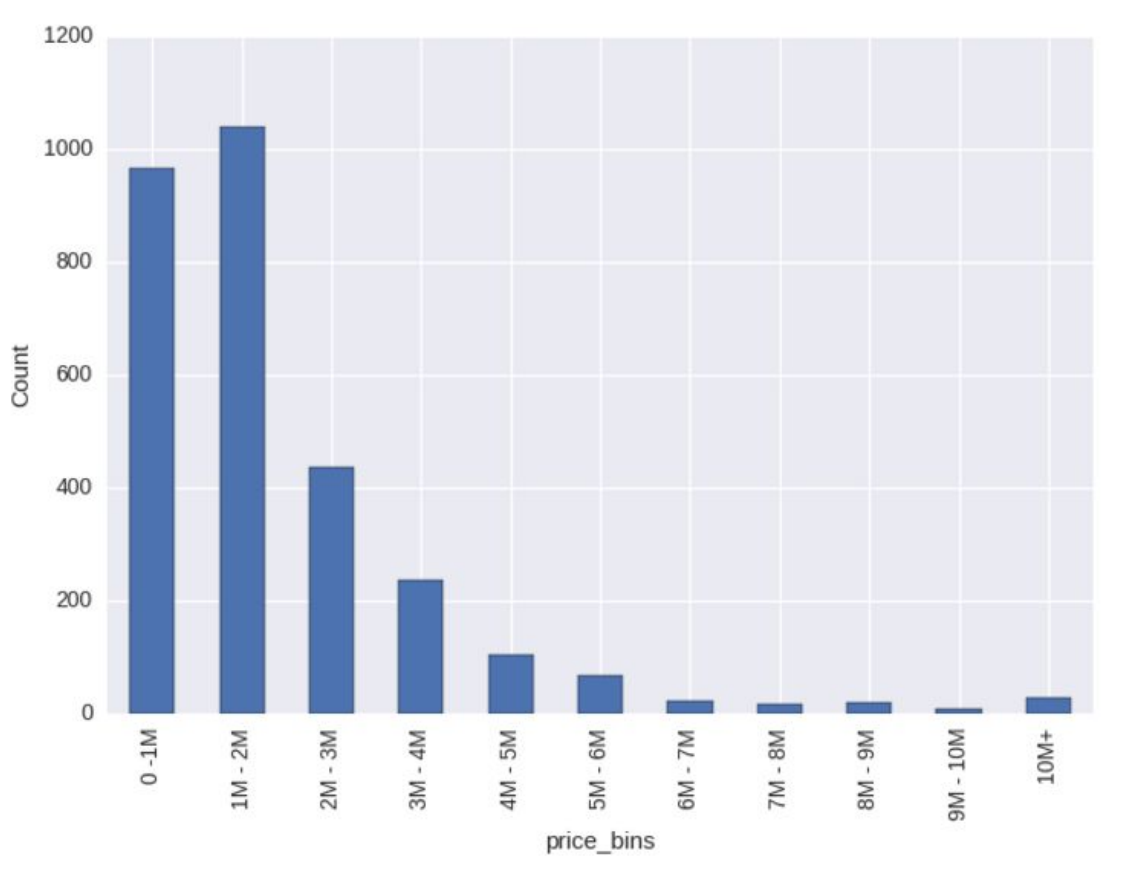
# Exploratory data analysis



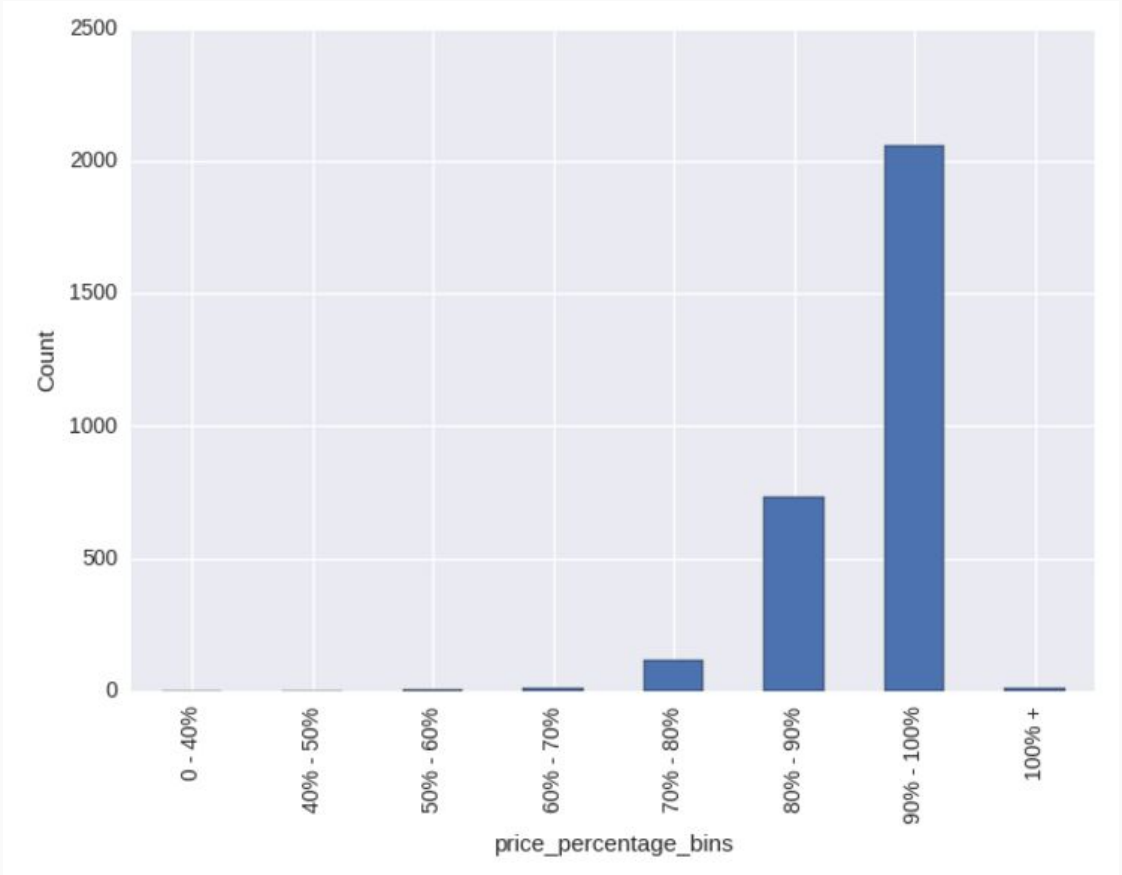
# Exploratory data analysis



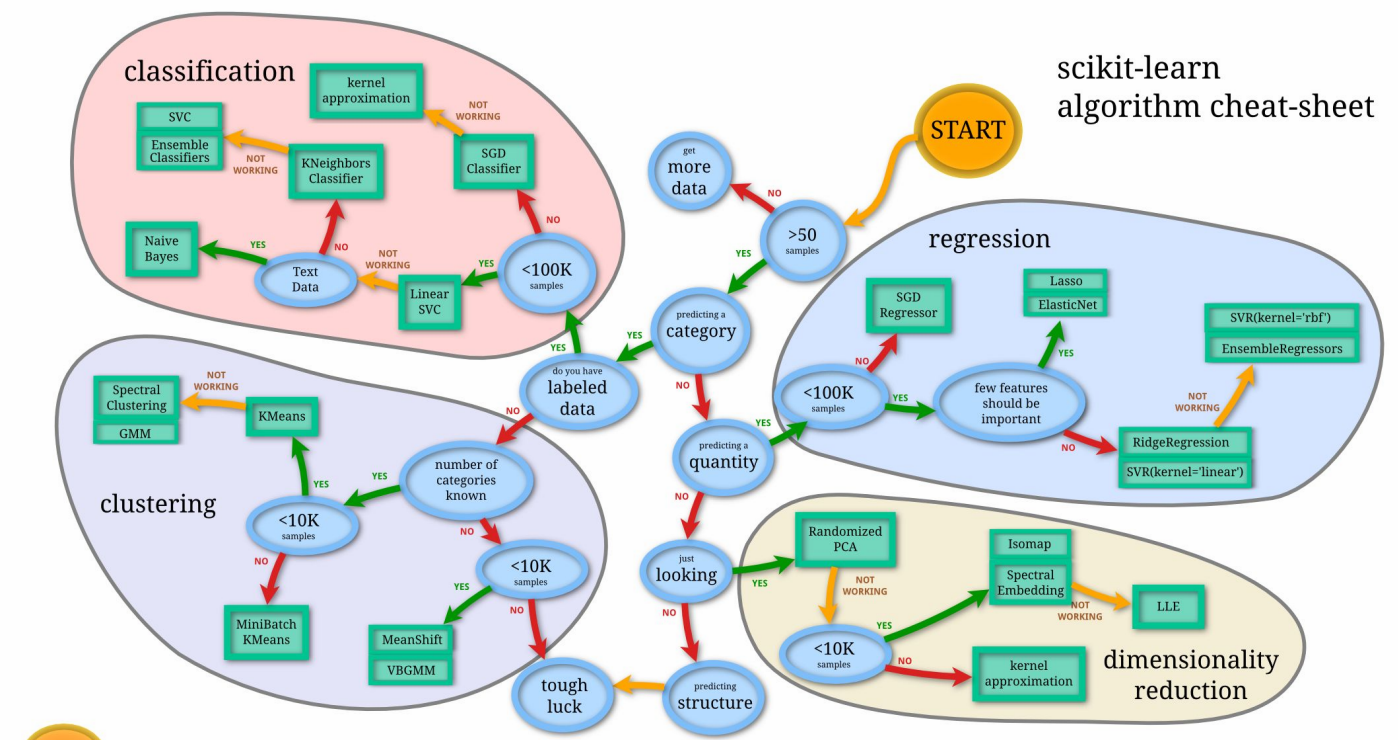
# Exploratory data analysis



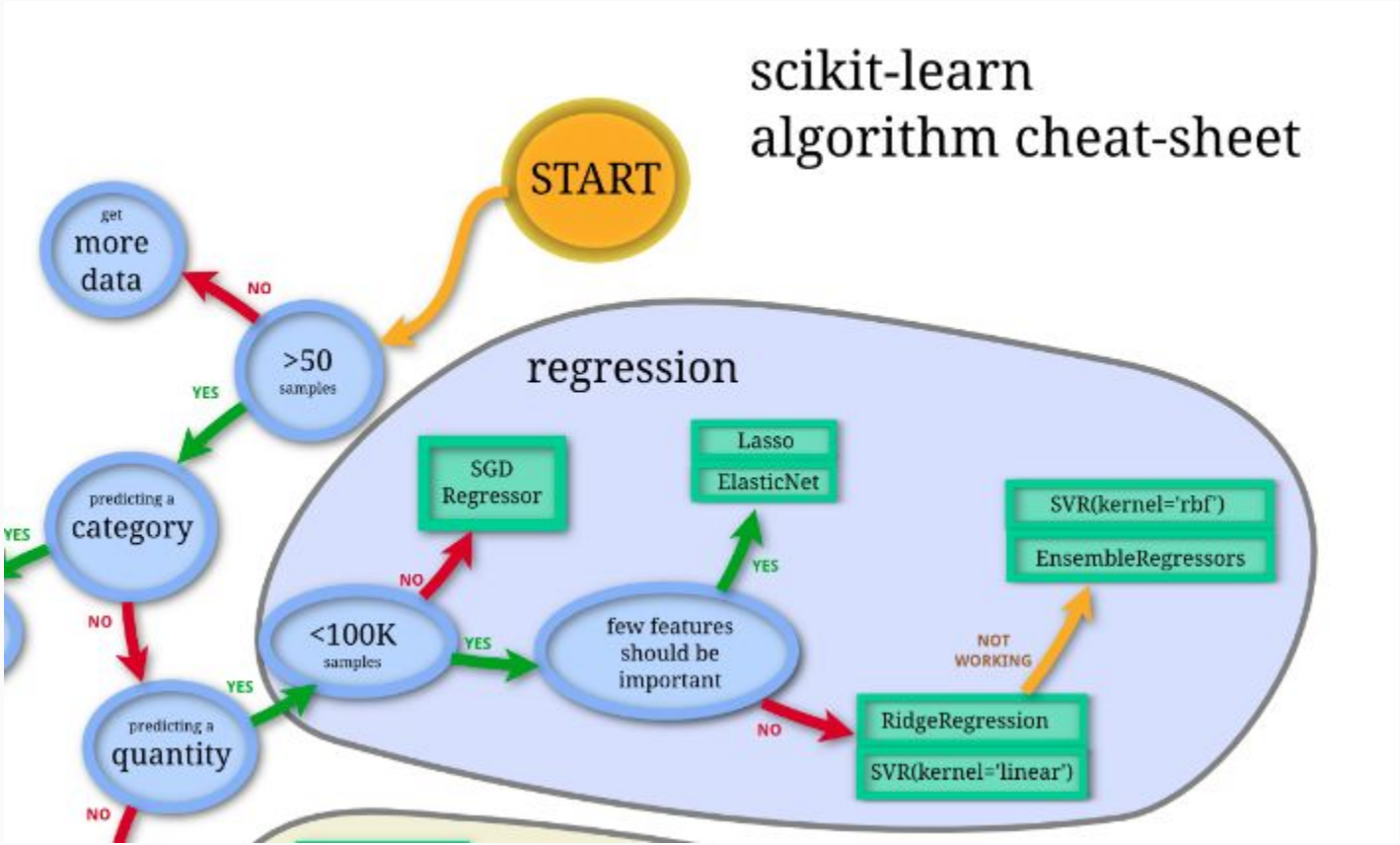
# Exploratory data analysis



# Model selection



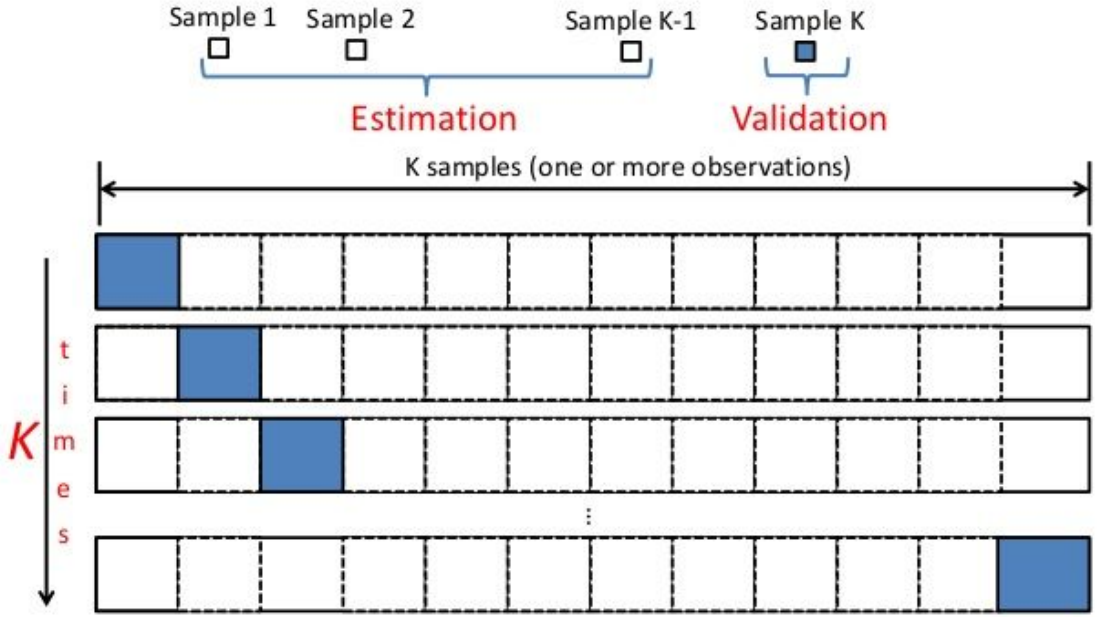
# Model selection



# Model selection

## Cross-validation: How it works?

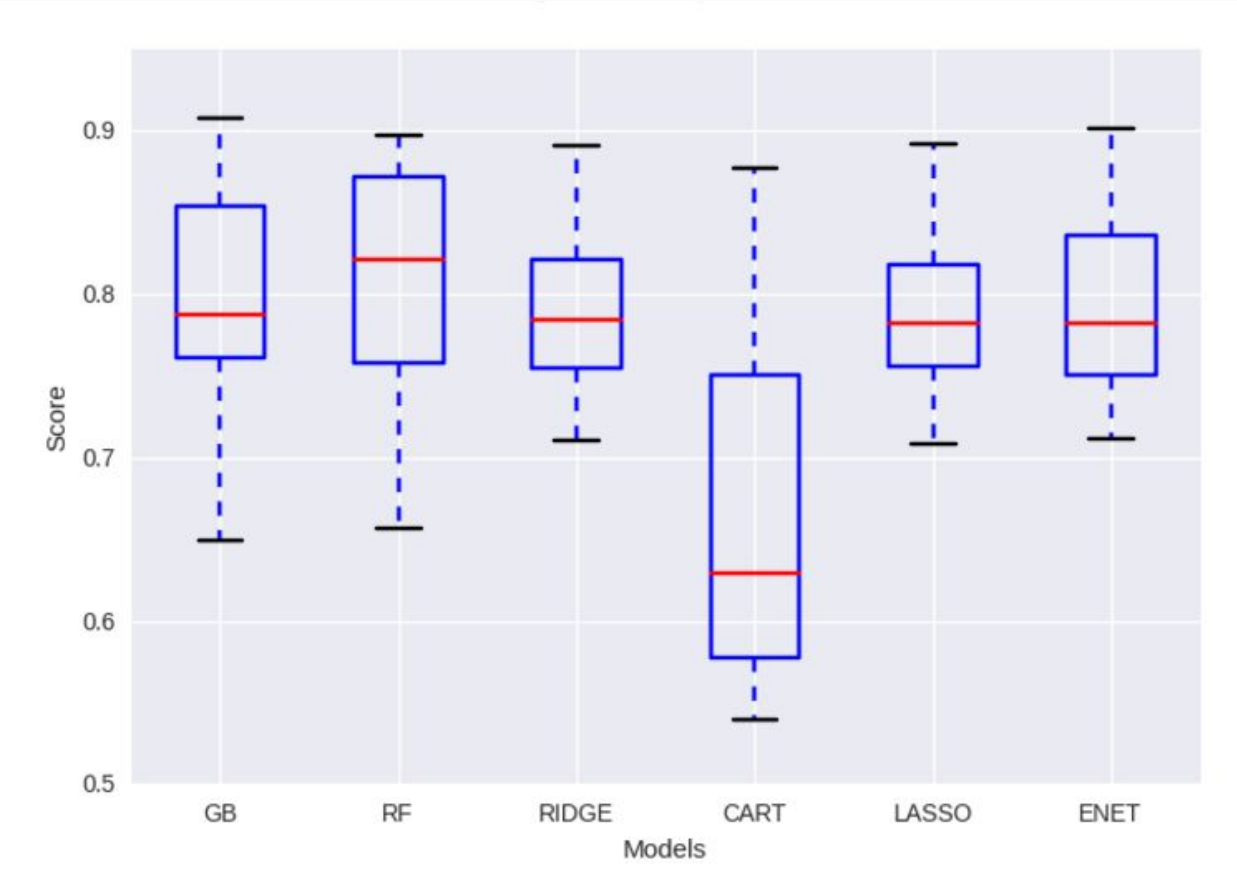
- K-fold cross-validation:



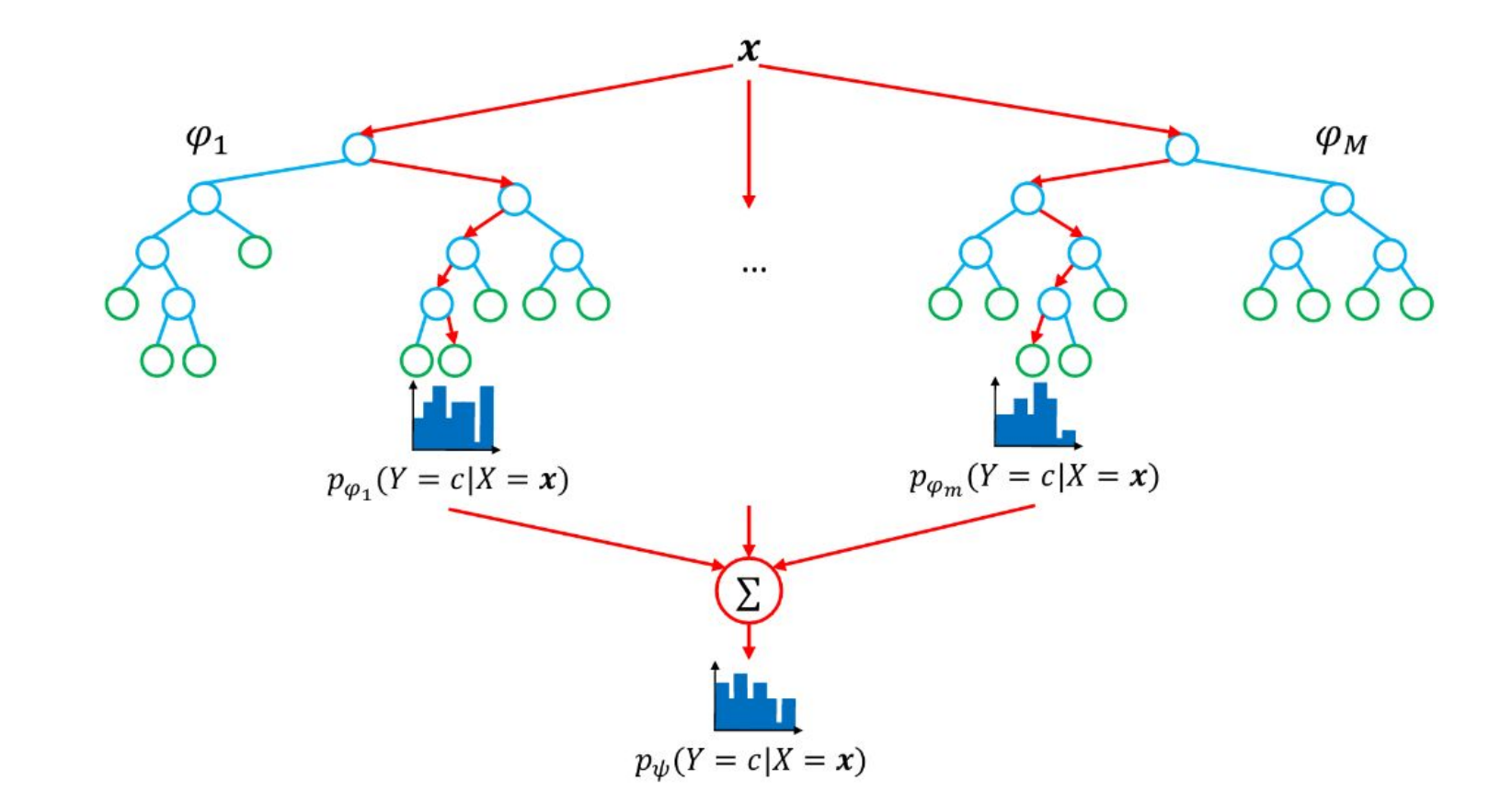
# Algorithm comparison

| Algorithm | Score | standard dev |
|---|---|---|
| Gradient Boosting | 0.785057 | 0.090029 |
| Random Forest | 0.803442 | 0.079728 |
| Ridge | 0.795482 | 0.056235 |
| CART | 0.671375 | 0.113846 |
| LASSO | 0.794522 | 0.056379 |
| ENET | 0.793873 | 0.059369 |

# Algorithm comparison



# Model selection: Random forest



# Model selection: Parameters tuning

The parameters of interest are

- N_estimators
  - The number of trees in the forest
- Max_depth
  - The number of questions to ask
- min_samples_leaf
  - The minimum size (number of sample) of a leaf

# Model selection: Parameters tuning

# Parameters tuning: Scorer

One can define a custom scorer function

```
# Import 'r2_score'
from sklearn.metrics import r2_score
def performance_metric(y_true, y_predict):
    """ Calculates and returns the performance score between
        true and predicted values based on the metric chosen. """
    # Calculate the performance score between 'y_true'and'y_predict'
    score = r2_score(y_true, y_predict)
    # Return the score
    return score


from sklearn.metrics import make_scorer
scoring_func = make_scorer(performance_metric, greater_is_better=True)
```

# Parameters tuning: Grid search

```
cv_sets = ShuffleSplit(X.shape[0], n_iter = 10, test_size = 0.20, random_state = 0)

# Create a random forest regressor object
regressor = ensemble.RandomForestRegressor(random_state = 0, n_jobs = -1)

# Create a dictionary for the parameters to search over
params = {
     'n_estimators': [10, 100, 200],
     'max_depth': range(1,20),
     'min_samples_leaf': [1,5,10]
  }
# Create the grid search object
grid = GridSearchCV(estimator=regressor, param_grid=params, scoring=scoring_fnc,
cv=cv_sets)

# Fit the grid search object to the data to compute the optimal model
grid = grid.fit(X, y)
```

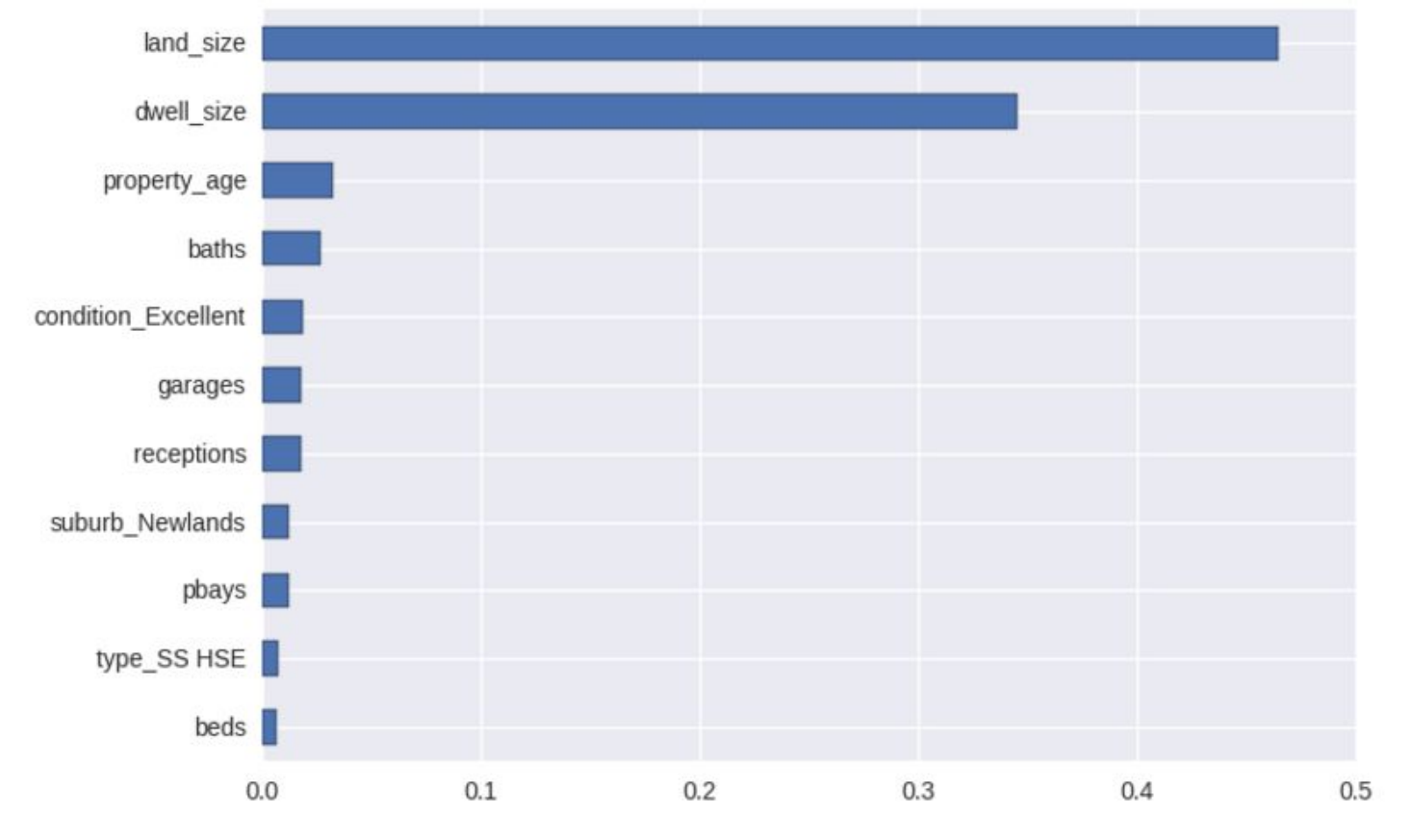
# Parameters tuning: Grid search

Best fit:

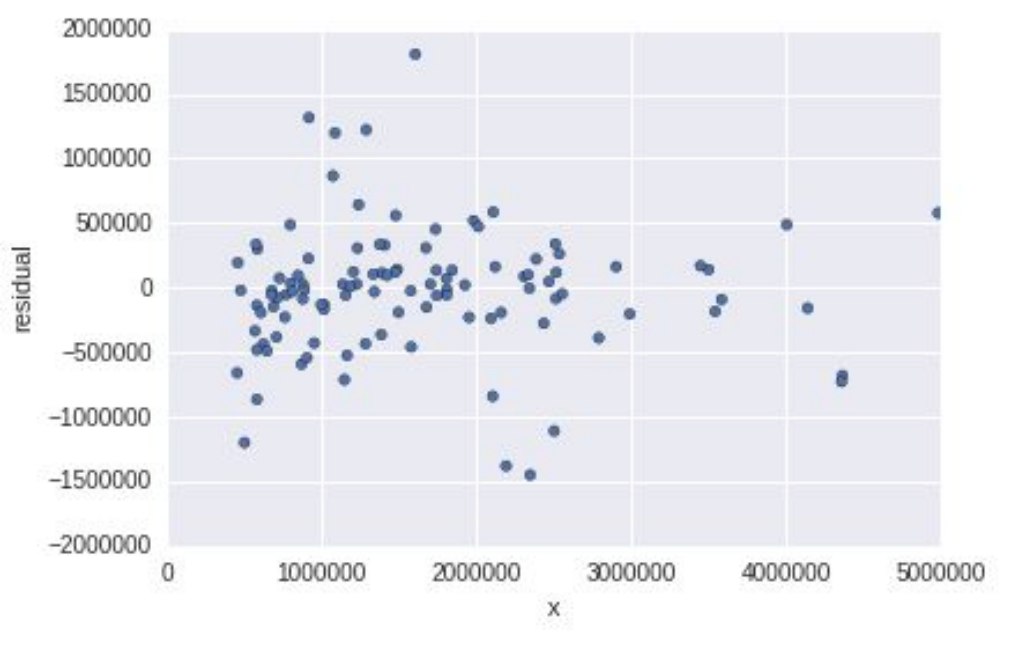
- `N_estimators 200`
- `Max_depth 17`
- `Min_samples_leaf 1`

```
R2 on training set: 0.80
R2 on test set: 0.73
```

# Model Evaluation: Feature importance



# Model evaluation: residuals



# Other considerations

For model selection

- Speed/effort vs accuracy
- Explanation vs accuracy

For improvement

- Outliers exclusion
- Consider other parameters

# Thank you

The ipython notebook:

https://github.com/alpha-ibrahim/Talks/tree/master/Talks-pydata_meetup