Machine Learning

| Evaluate Your System on the Test Set

Prof. Dr. Mostafa Elhosseini

https://youtube.com/@ProfElhosseiniTechVerse

Agenda

- Evaluate Your System on the Test Set

Remind

- Full Pipeline

```
from sklearn.model_selection import GridSearchCV

full_pipeline = Pipeline([
    ("preprocessing", preprocessing),
    ("random_forest", RandomForestRegressor(random_state=42)),
])
param_grid = [
    {'preprocessing__geo__n_clusters': [5, 8, 10],
     'random_forest__max_features': [4, 6, 8]},
    {'preprocessing__geo__n_clusters': [10, 15],
     'random_forest__max_features': [6, 8, 10]},
]
grid_search = GridSearchCV(full_pipeline, param_grid, cv=3,
                           scoring='neg_root_mean_squared_error')
grid_search.fit(housing, housing_labels)
```

Remind

- rnd_search

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

param_distribs = {'preprocessing__geo__n_clusters': randint(low=3, high=50),
                  'random_forest__max_features': randint(low=2, high=20)}

rnd_search = RandomizedSearchCV(
    full_pipeline, param_distributions=param_distribs, n_iter=10, cv=3,
    scoring='neg_root_mean_squared_error', random_state=42)

rnd_search.fit(housing, housing_labels)
```

Remind

- Final_model

```
final_model = rnd_search.best_estimator_  # includes preprocessing
feature_importances = final_model["random_forest"].feature_importances_
feature_importances.round(2)
```

Evaluate Your System on the Test Set

```
X_test = strat_test_set.drop("median_house_value", axis=1)
y_test = strat_test_set["median_house_value"].copy()

final_predictions = final_model.predict(X_test)

final_rmse = mean_squared_error(y_test, final_predictions, squared=False)
print(final_rmse)
```

```
41424.40026462184
```

Confidence Interval

- When you're thinking about replacing your current model with a new one that's just a tiny bit better, you want to be really sure it's actually going to be an improvement.
- Just seeing a statistic like "it's 0.1% better" isn't convincing enough to decide.
- That's where something called a "95% confidence interval" comes in handy

Confidence Interval

- Sampling distributions describe the variability in a sample statistic that occurs from sample to sample (or study to study)
- Sampling distributions help us to predict how close a statistic falls to the parameter it estimates
- We know that in a bell-shaped distribution, we expect to find nearly 100% of our values within 3 standard deviations of the mean

Confidence Interval

- A **confidence interval** is a range of values used in statistics to estimate the degree of uncertainty associated with a sample statistic.
- It gives an upper and lower limit for a parameter (like an average or percentage) and says, with a certain level of confidence (like 95%), that the true parameter value lies within this range.
- For example, if you have a 95% confidence interval for the average height of a group of people to be between 160 cm and 170 cm, it means you're 95% confident the true average height of the entire population falls within that range. It's a way to show how precise your estimate is and to account for the variability in your data.

Empirical Rule

- Approximately **68%** of the data falls within **1 standard deviation** (σ) of the mean (μ).
- Approximately **95%** of the data falls within **2 standard deviations** (2σ) of the mean.
- Approximately **99.7%** of the data falls within **3 standard deviations** (3σ) of the mean.
- The empirical rule allows you to quickly gauge **how dispersed data** is from the mean in a normal distribution. It is a useful tool for summarizing and interpreting data that follows an approximately normal pattern of distribution.

Confidence Interval

- Assume that a large number of independent measurements, all using the same procedure, are made on the diameter of a piston.
- The sample mean of the measurements is 14.0 cm, and the uncertainty in this quantity, which is the standard deviation of the sample mean, is 0.1 cm.
- Assume that the measurements are unbiased. The value 14.0 comes from a normal distribution, because it is the average of a large number of measurements.
- Now the true diameter of the piston will certainly not be exactly equal to the sample mean of 14.0 cm. However, because the sample mean comes from a normal distribution, we can use its standard deviation to determine how close it is likely to be to the true diameter.

Confidence Interval

- For example, it is very unlikely that the sample mean will differ from the true diameter by more than three standard deviations. Therefore we have a high level of confidence that the true diameter is in the interval (13.7, 14.3).
- It is not too unlikely for the sample mean to differ from the true value by more than one standard deviation. Therefore we have a lower level of confidence that the true diameter is in the interval (13.9, 14.1).
- The intervals (13.7, 14.3) and (13.9, 14.1) are confidence intervals for the true diameter of the piston
- we may be 99.7% confident that the true diameter of the piston is in the interval (13.7, 14.3), but only 68% confident that the true value is in the interval (13.9, 14.1).

Confidence Interval

- you can compute a 95% confidence interval for the generalization error using scipy.stats.t.interval().
- You get a fairly large interval from 39,275 to 43,467, and your previous point estimate of 41,424 is roughly in the middle of it:

```
from scipy import stats

confidence = 0.95
squared_errors = (final_predictions - y_test) ** 2
np.sqrt(stats.t.interval(confidence, len(squared_errors) - 1,
                         loc=squared_errors.mean(),
                         scale=stats.sem(squared_errors)))
```

```
array([39275.40861216, 43467.27680583])
```

Note

- If you **did a lot of hyperparameter tuning**, the performance will usually be slightly worse than what you measured using cross-validation.
- That’s because your system ends up fine-tuned to perform well on the validation data and will likely not perform as well on unknown datasets
- Story:
- Imagine you're training a robot to navigate through various mazes, and you keep tweaking its instructions (hyperparameters) to help it get better at navigating mazes you've shown it before (the validation data). By doing this a lot, the robot gets really good at these particular mazes but might not be as good at navigating new, unseen mazes because it's too specialized.

Project prelaunch phase

Present Your Solution:

- Share what you've created. Highlight key learnings, successes, and failures.
- What you learned throughout the project.
- What strategies or methods worked well and what didn't.
- Any assumptions you made during the project.
- The limitations or weaknesses of your system.

Project prelaunch phase

Document Everything:

- Keep detailed records of your work, including how the system was developed and any important decisions made along the way.

Project prelaunch phase

Create Clear Presentations:

- Make your findings easy to understand for everyone.
- Use clear visualizations to represent your data and results.
- Summarize key points with memorable statements, like how "median income is the top predictor of housing prices

Project prelaunch phase

Review Final Performance:

- Assess how well your system performs compared to experts in the field.
- In this example, the new system’s performance is close to that of expert estimations, which have a 30% error rate.
- Even if the improvement is small, the system might still be valuable if it saves experts time, allowing them to focus on more important tasks.