Ensemble Learning Part 2

19 March 2026

Alex Lyman

Ensemble Review

Ensembles

- Learning algorithms have different biases
  - They probably do not make the same mistakes
  - If one makes a mistake, the others may not
- Solution: model combination
  - Exploit variation in algorithms
    - Voting Ensemble, Stacking, Cascading, Delegating
  - Exploit variation in data
    - Bagging, Boosting, Random Forests

Four Important Criteria

- **Independence**: The various guesses have to be independent of one another. That is, each person must guess without the knowledge of what other people have guessed.
- **Diversity**: It is important to have a diverse set of guesses. In the guess the weight of the ox example, the people making the guesses ranged from farmers, butchers, livestock experts, housewives etc. That is, some people would be considered experts, while others would be considered as people with just a passing interest.
- **Decentralization**: The people making the guesses should be able to draw on their private, local knowledge.
- **Aggregation**: There must be some way of aggregating the guesses into a single collective guess. In the guess the weight of the ox example, this was done by taking the median guess. This is a common method, but others may also be used.

Variation in Algorithms

Variation in Algorithms

- If we use different models, we can combine them in lots of different ways to hopefully take advantage of their different weaknesses canceling out, but strengths compounding.
- Big difference – How to combine?
- Voting Ensemble
  - Everyone gets vote (hard or soft).
- Stacking
  - Train meta learner to weight models trained on data.
- Cascading
  - Pass from simpler model to harder model until high-confidence output.
- Delegating
  - Train router to predict which model would do best (not predict output) (like Mixture of Experts)

Variation in Data

Variation in Data

- Some models are really sensitive to jitters in training data.
- Training on things like **Bootstrap Samples** (sampled uniformly with replacement) and aggregating can get around that.
- That's called **Bagging.** (Bootstrap AGGregation)
- Random Forests – use both bootstrap samples *and* randomly select features to use, then aggregate small decision trees.

Variation in Data

- **Boosting** involves training several *weak learners* (learners that are better than chance, but not amazing), then ensembling them together.
- It might be a lot easier to train many slightly smart *weak learners* than one *strong learner*.
- Two boosting algorithms:
- Adaptive Boosting (AdaBoost)
- Gradient Boosting (xgBoost)

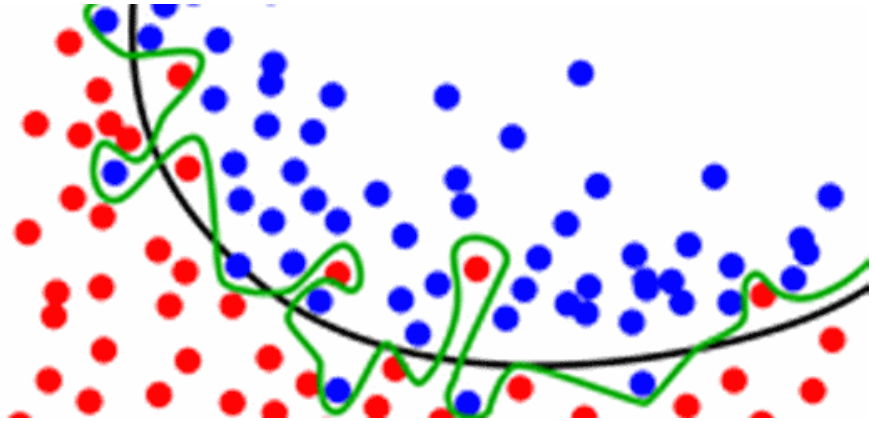
AdaBoost vs Gradient Boosting

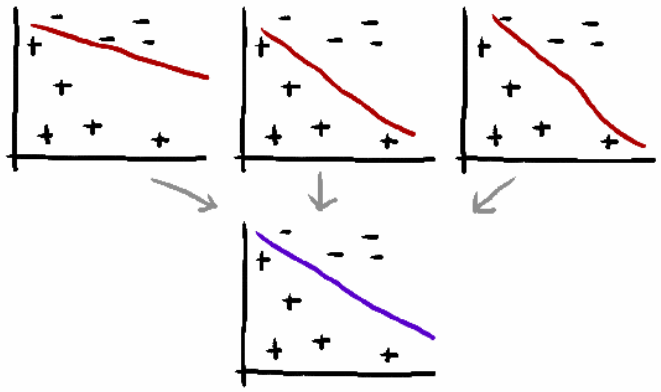
- **AdaBoost's Strategy:** "I missed that data point. I will make that data point physically larger/more important so the next tree is forced to look at it."
  - *Modifies the Data Weights*.
- **Gradient Boosting's Strategy:**
  - "I predicted the house costs $300k. The real price is $350k. My error is $50k. I will tell the next tree to completely ignore the house price, and just try to predict the number '$50k'."
  - *Modifies the Target Variable with residuals*.

Two New Slides

Averaging

- Train multiple models and average their results
- Often reduces overfit
  - Averaging multiple overfit models (green line) can move the ensemble towards the actual best model (black line)





Multi-level Ensemble

- You can further optimize scores by combining multiple ensembled models.
- ad-hoc approach: Use averaging, voting or rank averaging on manually-selected well-performing ensembles.
- Greedy approach: Start with a base ensemble of 3 or so good models. Add a model when it increases the train set score the most. By allowing put-back of models, a single model may be picked multiple times (weighing).
- Could also do a genetic algorithm search for the best combined model

Activity

Team Activity

- https://alexlyman.org/270/
- Download example notebook, train, and test data
- NO AI coding assistance – use sklearn docs.
- Turn off AI in colab.
- We'll work on this in teams for half an hour

CS 270

Resources for CS 270 Winter 2026

Instructor Information

Alex Lyman
TCMB 258
Office hours: By appointment

Competition Materials for March 19

- Competition Notebook
- Train Dataset
- Test Dataset

Competition.ipynb

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text

RAM Disk

Command palette Ctrl+Shift+P

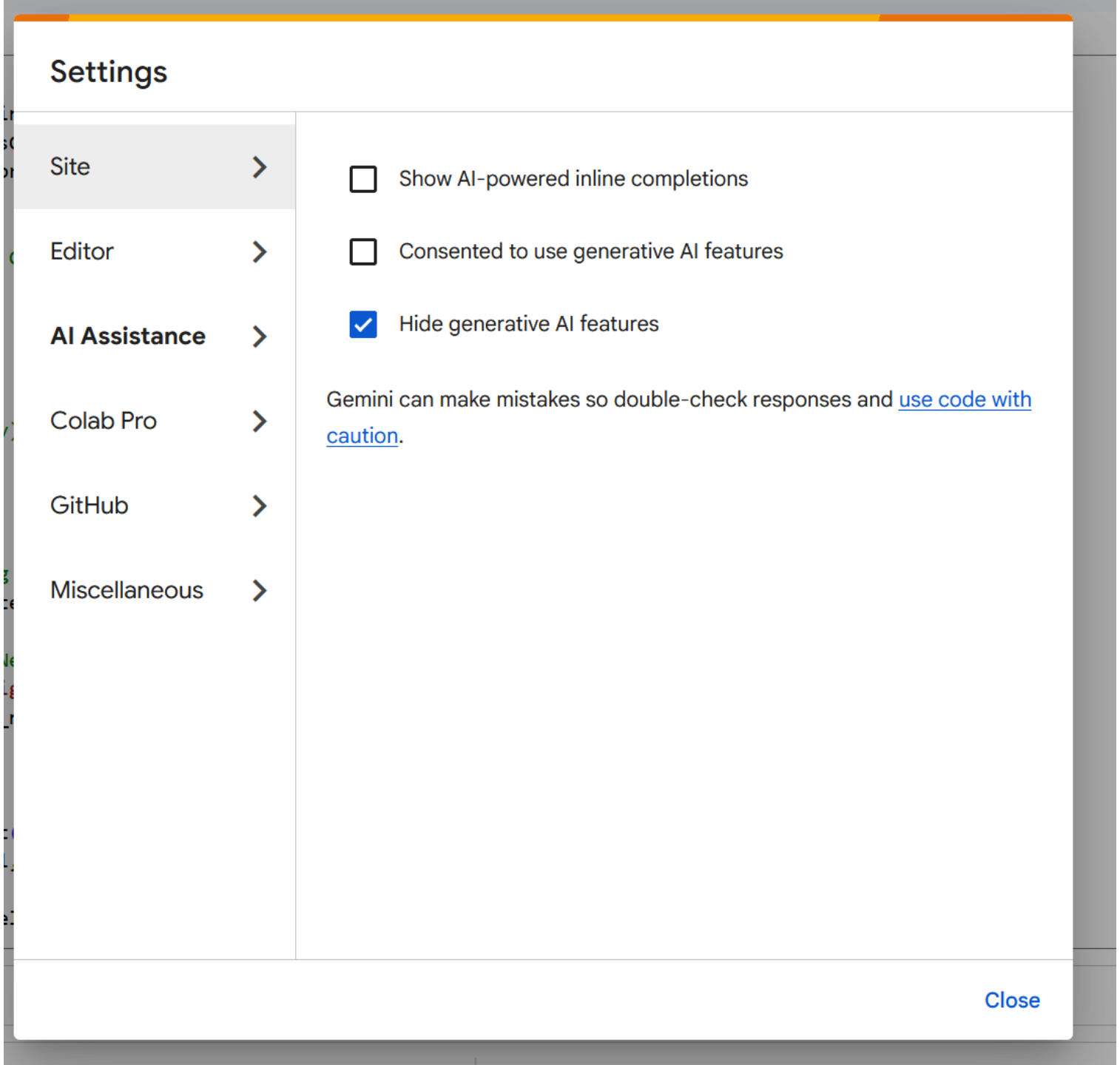
Settings

Keyboard shortcuts Ctrl+M H

Diff notebooks

[7]

```python
import pandas as
from sklearn.mode
from sklearn.neig
from sklearn.metr

# 1. Load the training data
# (Assuming the file is uploaded to the Colab environment)
df = pd.read_csv('kaggle_train.csv')

print("Data Overview")
display(df.head())

# 2. Separate features (X) and target (y)
X = df.drop(columns=['Target'])
y = df['Target']

# 3. Create a local Validation Set
# Test your ideas here before generating your final submission!
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Train a Baseline Model (K-Nearest Neighbors)
print("\nTraining Baseline K-Nearest Neighbors...")
baseline_model = KNeighborsClassifier(n_neighbors=3)
baseline_model.fit(X_train, y_train)

# 5. Evaluate the Baseline
val_predictions = baseline_model.predict(X_val)
baseline_accuracy = accuracy_score(y_val, val_predictions)

print(f"\nBaseline Model Accuracy: {baseline_accuracy * 100:.2f}%")
```


Settings
Site
Editor
AI Assistance
Colab Pro
GitHub
Miscellaneous
Show AI-powered inline completions
Consented to use generative AI features
Hide generative AI features
Gemini can make mistakes so double-check responses and use code with caution.
Close

Team Activity

- https://alexlyman.org/270/
- You have 25 minutes to work.
- Then, we'll see which team did best.
- Each team will explain what they did.
- Top 3 teams get a prize (that is hard to share, sorry).

- Try different things we've talked about during the semester.
- Don't use AI.
- I will not be watching, but God will.
- Use sklearn docs.
- Use code from earlier in the semester.