□ March ML Mania 2026

Your Participation Guide

Predict the NCAA Tournament — No Experience Required

□ Path A Basketball Fan	□ Path B Data Explorer	□ Path C Model Builder	□ Buddy Up Optional

□ Welcome!

This guide is everything you need to participate in the March ML Mania 2026 hackathon — whether you've never touched data in your life or you're a seasoned ML engineer.

There are no teams. No required meetings. No sign-up forms.
Just show up, pick a path, and go at your own pace.
You can work with a buddy informally if you want — or go completely solo. Either is great.

ABOUT THE COMPETITION — WHAT YOU'RE ACTUALLY DOING

March Machine Learning Mania is Kaggle's **12th annual** NCAA basketball prediction competition. It has a **$50,000 prize pool** and is one of the most beginner-friendly competitions on the platform — because everyone already understands the sport, and you can't "bust your bracket" since you predict every possible game, not just one path through the bracket.

□	□	□	□
$50K Prize Pool	**Deadline**	**Tournaments**	**12th**
Split across top finishers	March 19, 2026	Men's & Women's combined	Annual edition

What You're Predicting

You submit a probability — a number between 0 and 1 — for **every possible game matchup** in both the Men's and Women's NCAA tournaments. A probability of **0.7** means "I think Team A has a 70% chance of beating Team B." You're not picking one bracket path — you're putting a confidence level on every single possible game that could happen.

This means you can't bust your bracket. If you predict that a 12-seed has a 40% chance of beating a 5-seed and it happens — that's still a **good prediction**, because you gave it real odds. You're being **honest about uncertainty**, not picking winners and losers.

□ Men's AND Women's Tournaments — Both Count

This competition includes both the NCAA Men's and Women's tournaments in one combined submission file.

Men's team IDs start with a 1 (e.g., 1101). Women's team IDs start with a 3 (e.g., 3101).

You predict matchups for both. If you only know men's basketball, the AI tools guide will help you handle the women's side with historical data — or vice versa.

Stage 1 vs. Stage 2 — What's the Difference?

	Stage 1 — Before the Bracket	Stage 2 — After Selection Sunday
When	Now through ~March 18. The bracket hasn't been revealed yet.	March 15 (Selection Sunday) through March 19. The bracket is revealed and you can update your predictions with the actual matchups.
What	You predict probabilities for every theoretically possible matchup — thousands of team combinations. Think of it as predicting the whole universe of games that could happen.	You update your predictions knowing exactly which teams are in and which games are scheduled. You can fine-tune your numbers now that uncertainty is reduced.
Tip	Submit something in Stage 1 even if it's imperfect. You can update it. A submission on the board beats waiting forever for a perfect model.	This is your main chance to improve. Use your sports knowledge and the actual bracket to adjust predictions for teams you think are over or undervalued by the model.

How You're Scored — The Brier Score, Explained

This competition uses a metric called the **Brier Score**. It sounds technical but the idea is simple: **lower is better**, and it rewards you for being confident when you're right and penalizes you for being confident when you're wrong.

Brier Score in Plain English

Imagine you predict a game and say: 'I think Team A has an 80% chance of winning.'

SCENARIO A — Team A wins (you were right):
Your error = $(0.80 - 1.0)^2 = 0.04$ ← small error, good prediction

SCENARIO B — Team A loses (you were wrong):
Your error = $(0.80 - 0.0)^2 = 0.64$ ← big error, you were too confident

SCENARIO C — You hedged and said 50/50, Team A loses:
Your error = $(0.50 - 0.0)^2 = 0.25$ ← medium error, hedging protected you

Your final Brier Score is the average of these squared errors across ALL games.
Perfect score = 0.0 | Random guessing = 0.25 | Always wrong = 1.0

What This Means for Your Strategy

Unlike a traditional bracket where you're forced to pick winners, here you want to express your real uncertainty.

If you're very confident Team A wins — say 0.85 or higher
If it's a coin flip — say 0.50
If you think the upset is likely — say 0.35 for the favorite

Avoid saying 0.99 or 0.01 unless you're extremely sure — being wrong by that much hurts badly
Don't say 0.50 for everything either — you'll score 0.25, the same as random guessing

The sweet spot: be honest about your confidence level. Calibrated uncertainty beats overconfidence.

The Data You'll Work With

Kaggle provides decades of historical NCAA game data. Here's what the key files contain, in plain English:

File	What It Contains (plain English)
`MTeams.csv / WTeams.csv`	The master list of all teams and their ID numbers. Men's IDs start with 1, Women's with 3. You'll use these IDs to look up teams across every other file.

`MSeeds.csv / WSeeds.csv`	Which seed (1 through 16) each team received in each year's tournament. A '1' seed is the best team in their region. A '16' seed is the underdog. This is one of the most predictive features available.
`MNCAATourneyCompactResults.csv` `WNCAATourneyCompactResults.csv`	The result of every NCAA tournament game since 1985 — who played, who won, and by how many points. This is the foundation for training your prediction model.
`MRegularSeasonCompactResults.csv` `WRegularSeasonCompactResults.csv`	The result of every regular season game — same format as tournament results but covering thousands of games per year. Used to calculate win rates, point differentials, and Elo ratings.
`MNCAATourneyDetailedResults.csv` `WNCAATourneyDetailedResults.csv`	More detailed box score stats for tournament games: field goals, rebounds, turnovers, assists. Useful for building advanced features beyond just wins and losses.
`MNCAATourneySeedRoundResults.csv`	A summary of how each seed number has performed historically in each round. For example: how often does a 12-seed beat a 5-seed in Round 1? (Answer: about 35% of the time.)
`MTeamConferences.csv / WTeamConferences.csv`	Which conference each team belonged to in each year. Conferences like the ACC, Big Ten, and SEC tend to be stronger and produce more tournament success.

Where to Start with the Data

Don't try to use every file at once. Most strong submissions are built from just 3–4 files.

Recommended starting set: MSeeds.csv + MNCAATourneyCompactResults.csv + MRegularSeasonCompactResults.csv

Add MNCAATourneyDetailedResults.csv and MTeamConferences.csv once your baseline is working.

The AI prompts in Part 3 are designed around exactly this starting set.

What a Good Score Looks Like

Submission Type	Typical Brier Score	What It Means
Random (predict 50/50 every game)	**~0.250**	The floor. If you say 50% for everything, this is where you land.
Seed-only baseline (always pick the better seed)	**~0.220–0.230**	A simple model using just seeding. Your Path A first submission.
Good community submission	**~0.185–0.200**	Where solid Path B submissions typically land after Elo and win rate features.
Top 10% on Kaggle leaderboard	**~0.160–0.175**	Where advanced ensembles with external ratings (KenPom, Sagarin) compete.

Top 1% / Podium	**<0.155**	Requires strong calibration, external data, and ensemble blending.

A First-Timer's Realistic Goal

If you follow this guide and use the AI prompts, you can realistically land in the 0.190–0.215 range on your first attempt.

That puts you solidly ahead of random guessing and roughly in the range of ESPN's published bracket predictions.

For a first competition, that is genuinely impressive. Score improvement comes with iteration — and iteration comes from submitting early.

PART 1 — YOUR TIMELINE

Here is the full schedule from kickoff through the championship. Dip in and out as your life allows — even participating in one phase is worth it.

□ Dates	□ Phase	□ What To Do
Feb 27	**□ Kickoff**	• Join #march-ml-mania on Slack • Create a free Kaggle account at kaggle.com • Read this guide and pick your path — A, B, or C • Introduce yourself in the channel • Buddy up if you want: reply to someone else's intro
Mar 1–7	**□ Explore**	• Download the Kaggle data files and open the Starter Notebook • Run your first AI prompt from your chosen path (Part 3 of this guide) • Non-technical: research team strengths, upsets history, conference power • Technical: explore the dataset, run the starter notebook • Answer Week 1 Async Standup by Friday Mar 7
Mar 8–14	**⚙ Build**	• Make your first Kaggle submission — even a seed-based baseline counts • Run Prompts 2–3 from your path to improve your predictions • Share something in the channel — a chart, a score, an insight, anything • Answer Week 2 Async Standup by Friday Mar 14
Mar 15	**□ Selection Sunday!**	• Bracket is revealed — watch the channel for real-time posts • Update your predictions with the actual matchups • Submit your Stage 2 predictions before ~Mar 20 • Vote in the Final Four poll in the channel
Mar 19–30	**□ Tournament**	• Watch games and cheer (or groan) as your predictions play out • Post your reactions in #march-ml-mania • Answer Week 3 Async Standup • Check leaderboard updates posted in the channel
Apr 4–7	**□ Final Four**	• Follow the Final Four in the channel • Share your predicted champion • Cheer on the last games!
Apr 8–14	**□ Showcase**	• Share 1–3 things you learned — technical, sports, or personal • Post your final Kaggle score (no shame — every score counts) • Fill out the 3-question survey • You did a real ML competition. Celebrate that.

PART 2 — WHAT APPROACH FITS ME?

Pick the description below that sounds most like you. **You can mix approaches freely** — these are not boxes you're locked into. And if you want to work alongside someone informally, just say so in the channel.

The Basketball Fan

Best for: Sports fans, bracket fillers, anyone who watches college basketball

You know which teams are overseeded, who has a tough first-round draw, and which conferences have been dominating lately. That knowledge is real signal for predictions. Use AI to turn your intuition into data — no coding required.

→ Use Path A in Part 3 of this guide

The Data Explorer

Best for: Spreadsheet users, SQL folks, analysts, anyone comfortable with data even if not ML

You're comfortable reading data and asking questions of it. You don't need to know how a model works — you just describe what you want and your AI writes the code. You review the output and decide if it makes sense.

→ Use Path B in Part 3 of this guide

The Model Builder

Best for: Engineers, data scientists, Python or R coders who want to go deep

You want to actually build something — feature engineering, model training, calibration, ensemble blending. Use an AI coding agent as your co-pilot to move faster and explore ideas you might not have tried alone.

→ Use Path C in Part 3 of this guide

Want to Work with Someone?

This is fully individual — but informal collaboration is encouraged.

Just post in #march-ml-mania: 'Anyone want to work through this together?' or reply to someone else's intro.

No formal structure needed. A Slack DM and a shared Kaggle notebook is all it takes.

Getting Started on Kaggle (Everyone Does This First)

1. Go to kaggle.com and create a free account — takes 2 minutes
2. Search 'March Machine Learning Mania 2026' or use the link pinned in Slack
3. Click 'Join Competition' and agree to the rules (standard Kaggle terms)
4. Go to the Data tab and download all the CSV files as a zip
5. Unzip them to a folder on your computer — you'll give your AI tool access to this folder
6. The key files are:
   - MTeams.csv — list of all teams and their IDs

- MSeeds.csv — which seed each team received each year
- MNCAATourneyCompactResults.csv — every tournament game result since 1985
- MRegularSeasonCompactResults.csv — every regular season game result

7. Now go to Part 3 and run your first prompt ↓

□ Submission Format (Save This)

Your prediction file must be a CSV with exactly two columns:
ID — format: 2026_TeamA_TeamB (e.g., 2026_1101_1102)
Pred — a number between 0 and 1 (your predicted win probability for TeamA)

Common error: column names are case-sensitive. Use exactly 'ID' and 'Pred'.
You can submit up to 5 times per day — submit early and iterate.

PART 2 (CONT.) — RUNNING YOUR KAGGLE NOTEBOOK

Once you've downloaded the data, you need a place to run your code and generate your submission file. You have two options: **run directly on the Kaggle website** (easiest, no setup), or **run via the Kaggle API from your computer** (more control, great for Path B/C with an AI coding agent). Both options produce the same result — a **submission.csv** you upload to Kaggle.

Option 1 — Run Directly on the Kaggle Website (Recommended for Path A & B)

The fastest way to get started. Kaggle gives you free cloud compute right in your browser — no installation, no terminal, no local setup required.

8. Go to kaggle.com/competitions/march-machine-learning-mania-2026
9. Click the 'Code' tab at the top of the competition page
10. Click '+ New Notebook' — this opens a Python notebook in your browser
11. In the notebook: File → Add Data → Competition Data — the CSV files are now available
12. Write or paste your code (use the AI prompts from Part 3 to generate it)
13. Click 'Run All' to execute — results appear inline below each cell
14. When your submission.csv is ready: File → Save Version → Save & Run All
15. Click 'Submit to Competition' from the competition page to upload your output

Kaggle Notebooks Quick Reference

- Free CPU + GPU compute included — no account upgrade needed for this competition
- Notebooks auto-save; you can return and continue any time
- You can view other participants' public notebooks for inspiration: Code tab → Filter by 'Public'
- Official docs: kaggle.com/docs/notebooks

Option 2 — Run via the Kaggle API from Your Computer (Path B & C)

The Kaggle API lets you push notebooks from your computer to Kaggle's cloud and run them remotely — all from the command line or from within your AI coding agent. This is useful when you're iterating quickly and want to keep your code in a local folder.

Step 1 — Install the Kaggle Package

```
pip install kaggle
```

Run this once in your terminal. If you're inside your AI coding agent, just tell it: "Install the kaggle Python package."

Step 2 — Get Your API Token

16. Go to kaggle.com and click your profile picture → Settings
17. Scroll to the 'API' section and click 'Create New Token'

18. A file called kaggle.json will download — it looks like this:

```
{"username":"your-username","key":"your-api-key-here"}
```

19. Move this file to: ~/.kaggle/kaggle.json (Mac/Linux) or C:\Users\YOU\.kaggle\kaggle.json (Windows)
20. Set permissions so only you can read it (Mac/Linux only): chmod 600 ~/.kaggle/kaggle.json

Keep Your API Token Private

Never share your kaggle.json file or paste its contents anywhere.

Anyone with your API key can submit to competitions under your name and access your Kaggle account.

If it's ever exposed, go to Kaggle Settings → API → Expire Token and generate a new one.

Step 3 — Create a kernel-metadata.json in Your Project Folder

Every notebook you push to Kaggle needs a small config file that tells Kaggle how to run it. Create a file called kernel-metadata.json in your project folder:

```
{
  "id": "your-username/march-ml-mania-2026",
  "title": "March ML Mania 2026 — My Submission",
  "code_file": "solution.py",
  "language": "python",
  "kernel_type": "script",
  "is_private": true,
  "enable_gpu": false,
  "enable_internet": false,
  "dataset_sources": [],
  "competition_sources": ["march-machine-learning-mania-2026"],
  "kernel_sources": []
}
```

Replace your-username with your actual Kaggle username and solution.py with the name of your Python script.

Step 4 — Push and Monitor from Your Terminal

1	`kaggle kernels push -p ./my-project-folder` Pushes your folder to Kaggle and triggers a run. You'll see a URL in the output.
2	`kaggle kernels status your-username/march-ml-mania-2026` Check if your notebook is queued, running, or complete. Run this every 30–60 seconds.
3	`kaggle kernels output your-username/march-ml-mania-2026 -p ./output` Downloads your notebook's output files (including submission.csv) to a local folder.

Using AI coding agent to Automate Your Kaggle Workflow (Path C)

If you're in your AI coding agent, you can skip the terminal commands entirely and just describe what you want in plain English. Your AI coding agent handles the Kaggle API calls for you.

□	`Create a kernel-metadata.json file for my March ML Mania 2026 submission. My Kaggle username is [YOUR USERNAME]. The main script is solution.py. Keep it private, no GPU needed, and link it to the march-machine-learning-mania-2026 competition data.`
□	`Push the contents of this folder to Kaggle using the Kaggle API and monitor the run status every 30 seconds until it completes or fails. Show me the final status and any error messages.`
□	`My Kaggle kernel finished running. Download the output files to a folder called ./kaggle-output and show me what submission.csv looks like — the first 10 rows and the distribution of predictions.`
□	`Write a Python script that: (1) pushes my solution to Kaggle using the API, (2) polls the kernel status every 60 seconds, (3) downloads the output when complete, and (4) prints a summary of the submission file. Save it as push_and_monitor.py.`

□ **Learn More — Official Kaggle Resources**

Kaggle Notebooks documentation (all features, GPU, datasets, sharing):
→ kaggle.com/docs/notebooks

Community guide: Running Kaggle kernels from Python scripts via the API:
→ kaggle.com/discussions/getting-started/524433

Both links are worth bookmarking. The discussion guide has working code examples you can paste directly.

PART 3 — USING AI TO BUILD YOUR SUBMISSION

Your AI is your co-pilot, not a search engine. Describe what you want, and it writes the code, runs the analysis, and explains the results. You don't need to understand every line it produces — you just need to know what you're looking for and be able to tell it when something seems off.

The Core Idea

What matters is:

- You can describe what you want in plain English
- You can tell your AI when something looks wrong or confusing
- You can ask it to explain its output simply

That is a complete skill set for this competition. Nothing else required.

PATH A — THE BASKETBALL FAN

Best tool: Cowork (desktop app) or your AI of choice

"I follow March Madness every year. I know which teams are overseeded, which conferences are dominant, and which #5 seeds scare me. I want to turn that intuition into predictions."

Your workflow is entirely in plain English. You describe what you want, your AI produces it, you share the results in the channel. A written analysis or a chart is a legitimate contribution — no notebook required.

How to Start

21. Open Cowork and click 'Select Folder' — choose your Kaggle CSV folder
22. Or open your AI chat tool of choice and drag the CSV files directly into the chat
23. Copy and paste the prompts below one at a time — read each result before moving on
24. Ask your AI to explain anything you don't understand: 'What does this mean?'
25. Screenshot or copy results and share them in #march-ml-mania

Your 5 Prompts — Copy & Paste These

1	`I just downloaded the March ML Mania 2026 data from Kaggle. I have these files: MTeams.csv, MSeeds.csv, MNCAATourneyCompactResults.csv, and MRegularSeasonCompactResults.csv. Can you look at each file and give me a plain-English summary of what's in it and what it could be used for?`
2	`Using the tournament results file, which seed matchups have the biggest upsets? Show me the top 10 most common upset matchups (lower seed beating higher seed) as a simple table with win rates.`
3	`I want to understand which college basketball conferences are historically strongest in the tournament. Use the data to show me conference win rates in the NCAA tournament since 2010. Make a bar chart if you can.`
4	`I have no idea what 'Brier Score' means as a competition metric. Explain it to me like I'm not a math person, using a simple basketball game example. Tell me what a good score looks like versus a bad score for the March ML Mania 2026 competition.`
5	`Based only on historical seed matchup win rates, generate a prediction CSV file in the Kaggle format — columns 'ID' and 'Pred' — for all possible 2026 matchups. Use the historical win rate for that seed matchup as the prediction. Save it as my_submission.csv. This will be my first submission.`

Path A Tip

Your biggest contribution might not be a model at all.

A well-researched paragraph on why a specific team is overseeded this year — backed by stats your AI helped you pull — is real analysis.

Post it in the channel alongside your baseline submission. That is full participation.

PATH B — THE DATA EXPLORER

Best tool: Cowork (desktop app) — give it folder access to your CSVs

"I am comfortable with data and spreadsheets. I can follow code even if I don't write it from scratch."

Cowork writes and runs the Python for you. You guide the analysis, read the outputs, and decide what makes sense. You don't need to understand the code — just the results.

How to Start

26. Open Cowork and select the folder containing your Kaggle CSVs
27. Use Prompts 1–2 to explore the data and build your first feature table
28. Save the feature table as a CSV using Prompt 3
29. Use Prompt 5 to generate your Kaggle submission
30. Submit to Kaggle, get your score, then ask your AI what to improve next

Your 5 Prompts — Copy & Paste These

1	`I have the Kaggle March ML Mania 2026 CSVs in my selected folder. Please load MNCAATourneyCompactResults.csv and MSeeds.csv and show me: (a) how many tournament games are in the dataset, (b) the date range covered, and (c) a simple table showing win rates by seed number across all years.`
2	`Now load MRegularSeasonCompactResults.csv. For each team, calculate their average regular season win percentage and average point differential for the last 5 seasons (2021–2025). Save this as team_features.csv.`
3	`Add two columns to team_features.csv using MSeeds.csv: (a) each team's average tournament seed from the last 5 years, and (b) how many times they made the tournament in the last 5 years. Show me the top 25 teams by tournament appearances.`
4	`I want to add Elo ratings as a feature. Using regular season results, calculate a basic Elo rating for each team at the end of the 2025 season — start all teams at 1500, use K=20, update after each game. Add the final Elo to team_features.csv.`
5	`Using team_features.csv, build a logistic regression model to predict the win probability for each possible 2026 matchup, using seed difference and Elo difference as features. Train it on tournament games from 2016–2024, generate predictions for all 2026 matchups, and save the output as submission.csv with columns 'ID' (format: 2026_TeamA_TeamB) and 'Pred'.`

Path B Tip

After Prompt 5 your Brier Score will likely be around 0.210–0.230 — already beating random guessing (0.250) and the average bracket picker.

The biggest gains come from better features, not a different model.
Ask your AI: 'What features are most predictive in NCAA tournament models?' and iterate from there.

□ PATH C — THE MODEL BUILDER

Best tools: AI coding agent (terminal) or Cowork with a script-first workflow

"I write Python or R. I want to build a real model — feature engineering, calibration, ensemble blending — and use an AI coding agent to move faster."

Your workflow is iterative model development. Use an AI coding agent to generate code you review and modify — not just run blindly. The prompts below take you from raw data to a competitive submission, then give you clear upgrade paths.

How to Start

31. Open a terminal in your project folder (with your Kaggle CSVs) and launch your AI coding agent
32. Use Prompt 1 to bootstrap a full end-to-end pipeline
33. Submit to Kaggle immediately to get your baseline Brier Score
34. Use Prompts 2–4 to add features and improve the model one layer at a time
35. After each change, submit and track your score in a simple notes file

Your 5 Prompts — Copy & Paste These

1	`I have the March ML Mania 2026 Kaggle data in this directory. Build a complete end-to-end Python script that: (1) loads MSeeds.csv, MNCAATourneyCompactResults.csv, and MRegularSeasonCompactResults.csv, (2) engineers features including seed difference, historical seed matchup win rate, and 5-year regular season win percentage, (3) trains a logistic regression on tournament results from 2010–2024, (4) generates predictions for all possible 2026 matchups, and (5) saves submission.csv with columns ID and Pred. Include probability calibration so predictions don't cluster near 0.5.`
2	`Improve the model by adding Elo ratings. Calculate game-by-game Elo for every team using regular season and tournament results going back to 2003. Use K=32 for tournament games and K=20 for regular season. Add each team's end-of-season Elo as a feature, retrain, and compare Brier Score on a 2022–2024 holdout vs. the baseline.`
3	`Add three more features: (a) strength of schedule — average opponent win percentage, (b) recency-weighted win rate giving 3x weight to the last 10 games, and (c) coach tournament experience — total tournament wins by that team's coach in the dataset. Then use SHAP or feature importance to show which features drive predictions most.`
4	`Train three models — logistic regression, XGBoost, and a simple 2-layer neural net (64 units) — on the same feature set. Blend predictions with equal weights and also try optimizing blend weights using scipy minimize on the holdout set. Report Brier Score for each model and both blends.`
5	`My Kaggle submission is giving this error: [PASTE ERROR]. Here is the relevant code: [PASTE CODE]. Diagnose the issue, fix it, and add a validation step that checks the submission CSV for common Kaggle format errors before I upload.`

Path C Tip

The top competitors typically blend: Elo ratings, KenPom advanced stats, and a model trained specifically on tournament games.

The single biggest lever is probability calibration. Uncalibrated models get punished hard by the Brier Score.

Ask your AI: 'Apply Platt scaling and show me the calibration curve.' This alone can drop your score significantly.

PART 3 (CONT.) — PROMPTS FOR EVERYONE, ANY TIME

Use these at any point when you're stuck, confused, or just want to share what you've done more clearly.

□	I got this Kaggle score: [PASTE SCORE]. What does this mean compared to a random guesser (Brier Score ~0.250), a seed-only baseline (~0.220–0.230), and a competitive top-10% submission (~0.160–0.175) for the March ML Mania 2026 competition?
□	I don't understand this part of the data: [PASTE CONFUSING COLUMN OR ROW]. Can you explain what it represents in plain language and give me a quick example of how it would be used in a prediction?
□	Here is my current approach: [DESCRIBE IT IN 2–3 SENTENCES]. What are the three most likely reasons my predictions are wrong, and what is the simplest fix for each one?
□	I want to share what I did in the Slack channel but I can't explain it clearly. Can you write a 3–4 sentence plain-English summary of this approach: [PASTE YOUR CODE OR DESCRIBE YOUR METHOD]?

PART 4 — PREDICTION STRATEGIES & MODEL PLAYBOOK

This section is your reference guide for **how to actually build your predictions** — from the simplest one-liner approach all the way to competitive ML models. Everything here can be generated by your AI or coding agent. You don't need to write the code yourself. You just need to understand what you're asking for and why.

The Prediction Ladder — Pick Your Level

Every approach below is a legitimate competition entry. The ladder is not a checklist — pick the level that matches your time and interest, build something that works, and submit it.

Approach	Difficulty	Best For	Typical Score	Time
□ **Historical Seed Win Rates**	Beginner	**Path A**	**~0.220–0.230**	30 min
□□ **Elo Ratings**	Intermediate	**Path A/B**	**~0.205–0.220**	1–2 hrs
□□ **Logistic Regression + Features**	Intermediate	**Path B**	**~0.190–0.210**	2–3 hrs
□□□ **XGBoost / Random Forest**	Advanced	**Path C**	**~0.178–0.195**	3–5 hrs
□□□□ **Ensemble + Calibration**	Expert	**Path C**	**~0.160–0.180**	5+ hrs

□ Approach 1 — Historical Seed Win Rates (Path A)

What it is: Look at every NCAA tournament game played since 1985. For each possible seed matchup (e.g., 1-seed vs. 16-seed, 5-seed vs. 12-seed), count how many times the higher seed won and divide by total games played. Use those historical rates as your predictions.

Why it works: Seeds are assigned by the selection committee specifically to reflect team strength. A 1-seed has won their first-round matchup roughly 99% of the time. A 12-seed beats a 5-seed about 35% of the time. These patterns hold year over year because seeding is consistent.

Limitation: It treats every team in a given seed the same. A 3-seed might be Duke or a mid-major that scraped in — the model doesn't know the difference. This is where more advanced approaches improve.

□ Prompt to Generate It (for your AI of choice, Cowork, or AI coding agent)

I have MSeeds.csv and MNCAATourneyCompactResults.csv from the March ML Mania 2026 dataset.

For every possible seed matchup (1v16, 2v15, ... 8v9, and cross-region matchups), calculate the historical win rate of the higher seed across all years in the dataset.

Then generate a submission.csv where every 2026 matchup prediction uses that seed matchup win rate as the probability. Format: columns ID (e.g., 2026_1101_1145) and Pred (probability between 0 and 1).

Show me a table of the 10 biggest upset seed matchups (lowest win rate for the better seed) before saving the file.

□ Approach 2 — Elo Ratings (Path A/B)

What it is: Elo is a rating system invented for chess and adapted for sports. Every team starts with a rating (usually 1500). After each game, the winner gains points and the loser loses points — but the amount exchanged depends on how surprising the result was. Beating a #1 team earns you more than beating a weak opponent. Over a full season, good teams climb to ~1700+ and weak teams fall to ~1200.

Why it works: Elo captures momentum and schedule quality in ways that simple win-loss records miss. A team that beat three top-10 opponents and lost to a weak team looks very different from a team with the same record that played an easy schedule. That difference matters a lot in tournament predictions.

Key parameters: K-factor controls how fast ratings change. Use K=20 for regular season games, K=32 for tournament games (upsets matter more). Starting Elo = 1500. Apply a home-court adjustment of ~100 points.

□ Prompt to Generate It

Using MRegularSeasonCompactResults.csv and MNCAATourneyCompactResults.csv, build an Elo rating system for every NCAA team.

Rules: start all teams at 1500, K=20 for regular season, K=32 for tournament games, add 100 to the home team's effective Elo before each game.

Process all games chronologically from 2003 to end of 2025 regular season. Return the final Elo rating for every team that appears in MSeeds.csv for 2026.

Show me the top 20 and bottom 20 teams by Elo, and save the result as elo_ratings.csv with columns TeamID and Elo.

□ Approach 3 — Logistic Regression (Path B)

What it is: Logistic regression is a simple machine learning model that takes numeric features (like seed difference, Elo difference, win rate) and learns how much weight to give each one to predict a probability. It draws the best possible straight-line boundary between wins and losses in feature space.

Why it works: It's fast, transparent, and well-calibrated out of the box. You can see exactly which features matter most by looking at the coefficients. For March Madness, it consistently performs well because the relationship between features and outcomes is relatively linear.

Best features to include (in order of predictive power):

Feature	Plain-English Meaning
Seed difference	Higher seed - lower seed. A 1v16 matchup = difference of 15. Very strong signal.
Elo difference	Team A Elo minus Team B Elo at end of regular season. Captures performance quality.
Win rate (last 5 seasons)	How often did each team win? Averaged over recent years to reduce noise.
Point differential	Average margin of victory. Beating opponents by 20 vs. 2 matters for predictions.
Tournament appearances (last 5 yrs)	Experience matters. Teams that consistently make the field handle pressure better.
Conference strength	ACC, Big Ten, SEC teams face tougher schedules — their records deserve more credit.
Seed matchup historical win rate	Cross-reference: how often has a team seeded X beaten a team seeded Y historically?

□ Prompt to Generate It

Using team_features.csv (which has Elo, win rate, seed, point differential, conference), train a logistic regression model to predict NCAA tournament game outcomes.

Training set: all tournament games from 2010–2024 in MNCAATourneyCompactResults.csv. For each game, compute the difference in each feature between Team A and Team B.

Validate on 2022–2024 holdout (do not train on these years). Report the Brier Score on the holdout.

Apply Platt scaling (isotonic calibration) to the output probabilities. Show the calibration curve before and after.

Then generate predictions for all possible 2026 matchups and save as submission_logistic.csv.

□ Approach 4 — XGBoost / Gradient Boosting (Path C)

What it is: XGBoost (Extreme Gradient Boosting) is a powerful ML model that builds hundreds of small decision trees, each one learning from the mistakes of the previous tree. The final prediction is a weighted sum of all those trees. It handles non-linear relationships (e.g., Elo only matters above a certain threshold) and feature interactions that logistic regression misses.

Why it wins competitions: XGBoost won Kaggle's first March Mania competitions outright and continues to be a core component of top ensembles. It's particularly good when you have many features and the relationships aren't purely linear — which is true in basketball, where context matters a lot.

Watch out for: Overfitting. XGBoost will memorize the training data if you let it. Always validate on a holdout set you've never trained on (use 2022–2024 for testing). Limit tree depth to 3–5 and use early stopping.

□ Prompt to Generate It

Train an XGBoost classifier on the same feature set as the logistic regression (seed diff, Elo diff, win rate diff, point diff, conference strength). Use the same 2010–2021 training period and 2022–2024 holdout.

Settings to start: max_depth=3, n_estimators=200, learning_rate=0.05, subsample=0.8, colsample_bytree=0.8.

Use early stopping on the holdout set (stop if Brier Score doesn't improve for 20 rounds).

Plot feature importance using SHAP values — show me which features are driving predictions most.

Apply isotonic regression calibration to the output probabilities and compare Brier Score before and after calibration.

Save predictions as submission_xgb.csv.

□ Approach 5 — Ensemble Blending + Calibration (Path C)

What it is: Combine predictions from multiple models into a single submission. For example: take your logistic regression's prediction, your XGBoost prediction, and an Elo-only baseline prediction — then average them (or find the optimal weighted average). This almost always beats any single model on its own.

Why it works: Different models make different errors. When one model is overconfident on a matchup, another might be more measured. By averaging, you smooth out individual model errors. Top Kaggle solutions almost always use ensembles.

□ Prompt to Build the Ensemble

I have three prediction files: submission_logistic.csv, submission_xgb.csv, and submission_elo_baseline.csv — each with columns ID and Pred.

Step 1: Simple blend — average the three predictions for each ID and save as submission_blend_equal.csv.

Step 2: Optimized blend — use scipy.optimize.minimize to find weights (w1, w2, w3 that sum to 1.0) that minimize Brier Score on the 2022–2024 holdout. Save as submission_blend_optimized.csv.

Step 3: Apply isotonic regression calibration to the optimized blend and show the calibration curve.

Report Brier Score on the holdout for each file: individual models, equal blend, optimized blend, calibrated blend.

Calibration — The Hidden Multiplier

Calibration means: when your model says 70%, does Team A actually win about 70% of the time?

Most models are overconfident — they say 85% when they mean 70%. The Brier Score punishes this hard.

Platt scaling and isotonic regression are two methods that 'squish' predictions toward the true frequency.

Ask your AI: 'Show me a reliability diagram (calibration curve) for my model predictions vs. actual outcomes.'

If your curve is above the diagonal, you're overconfident. Fix it before submitting.

PART 4 (CONT.) — DATA EXPLORATION PROMPTS

Use these prompts in any tool — Cowork, your AI of choice, or AI coding agent — to understand the data before you build anything. Good exploration catches problems early and reveals which features are worth building.

□	Load MNCAATourneyCompactResults.csv and give me a full EDA (exploratory data analysis): how many games, years covered, unique teams, wins by seed, average margin of victory by round, and any missing values. Show me the 5 biggest upsets by point differential.
□	From MRegularSeasonCompactResults.csv, show me: (1) which teams played the most games, (2) which teams had the highest win rate over the last 5 seasons, (3) which teams had the most consistent performance (lowest year-over-year variance in win rate), and (4) which conferences had the highest average win rates.
□	Load MSeeds.csv and show me: how many unique teams appeared in the tournament per year, the distribution of seeds (how many 1-seeds, 2-seeds, etc.), and which teams have the most tournament appearances in the last 10 years. Make a bar chart of appearances by seed.
□	I want to understand correlations. For tournament games, compute: correlation between seed difference and win probability, between point differential (regular season) and win probability, and between Elo difference and win probability. Which is the strongest predictor? Show scatter plots for each.
□	Show me which features are most different between teams that win in the first round vs. teams that lose. Compare: average seed, average regular season win rate, average point differential, and average Elo. What stands out?
□	Are there any systematic patterns in upsets? For 5v12 and 6v11 matchups (known upset-prone seed lines), which conferences produce the most upsets? Which years had the most upsets? Does conference strength predict upset likelihood?

⚙ Feature Engineering Prompts

Feature engineering means creating new columns from raw data that help the model understand patterns better. These prompts build the most impactful features for NCAA tournament prediction.

⚙	Using MRegularSeasonCompactResults.csv, calculate for each team in the 2026 tournament: (a) 5-year average win rate, (b) 5-year average point differential, (c) win rate in last 10 games of the regular season, (d) win rate against teams with a win rate above 60%, and (e) home vs. away win rate split. Save as team_features.csv.
⚙	Add a 'strength of schedule' column to team_features.csv. For each team, calculate the average win rate of all their opponents in the 2025 regular season. A team that beat 20 opponents with a 65% average win

	rate has a harder schedule than one who beat 20 teams averaging 45%. Weight by number of games played against each opponent.
⚙	Create a 'tournament experience' feature. For each 2026 team, count: total tournament wins in the last 10 years, average seed in the last 5 tournament appearances, and number of Final Four appearances in the last 10 years. Add these to team_features.csv.
⚙	Using MTeamConferences.csv and regular season results, calculate conference-level strength for each year: the average win rate of conference teams when playing against teams from other conferences. Assign each team their 2025 conference strength score and add it to team_features.csv.
⚙	Create a recency-weighted win rate: give each game a weight based on how recent it is (games from 2025 get weight 1.0, games from 2024 get 0.7, from 2023 get 0.5, from 2022 get 0.3, older get 0.1). Compute weighted win rate for each team and add to team_features.csv. Compare this to the simple 5-year average — which teams change rank significantly?

Model Building & Evaluation Prompts

Use these prompts to train models, evaluate them honestly, and compare approaches. Always use a holdout set you did not train on — 2022–2024 is recommended.

```
Build a complete ML pipeline for March ML Mania 2026. Load
team_features.csv and MNCAATourneyCompactResults.csv. For each
historical tournament game, create a row with features: [Team A Elo,
Team B Elo, Team A seed, Team B seed, Team A 5yr win rate, Team B 5yr
win rate, Team A point diff, Team B point diff, Team A SOS, Team B SOS].
Target = 1 if Team A wins. Train logistic regression on 2010–2021,
evaluate on 2022–2024 holdout. Show Brier Score, ROC-AUC, and a
calibration curve.
```

```
I want to compare three models side-by-side on the same features and
holdout: logistic regression, random forest (100 trees, max_depth=5),
and XGBoost (200 trees, lr=0.05, max_depth=3). For each: report training
Brier Score, holdout Brier Score, and whether it's overfitting. Plot a
bar chart comparing holdout scores.
```

```
My model's calibration curve shows it's overconfident — predictions
cluster around 0.3 and 0.7 when they should be more spread. Apply three
calibration methods: Platt scaling (logistic regression on outputs),
isotonic regression, and temperature scaling. Show the calibration curve
for each method and report which one gives the best Brier Score on the
holdout.
```

```
I want to understand what's driving my model's mistakes. For my 2022–
2024 holdout predictions, find the 20 games where my model was most
wrong (largest squared error). Show me the actual matchup details for
each: teams, seeds, game result, my predicted probability. Is there a
pattern in where I'm losing points?
```

```
Run a cross-validation experiment across years: train on all years
except 2022, test on 2022. Then train on all except 2023, test on 2023.
Then all except 2024, test on 2024. Report Brier Score for each year and
the average. This tells me if my model is consistent across different
tournament conditions.
```

```
Generate my final 2026 submission. Use all tournament games from 2010–
2025 for training (include the holdout years now). Apply calibration.
Generate probabilities for all possible 2026 matchups using the
SampleSubmission.csv as the template for matchup IDs. Validate: check
that all IDs match, all predictions are between 0 and 1, and no missing
values. Save as final_submission.csv.
```

End-to-End Agent Workflow

If you want to hand your AI coding agent (or Cowork) the entire pipeline in one session, use this sequence. Each prompt builds on the previous one. Copy them in order.

The Full Session — Run These Prompts in Order

SESSION GOAL: Go from raw Kaggle data to a validated, calibrated submission file.

1. DATA CHECK: Load all CSV files in this folder. Tell me: how many rows each has, what years are covered, which team IDs appear in MSeeds.csv for 2026, and whether there are any missing values.

2. FEATURE BUILD: Calculate for every 2026 tournament team: Elo rating (all games since 2003), 5-year win rate, 5-year point differential, strength of schedule, and tournament experience. Save as team_features.csv.

3. PIPELINE BUILD: Train a logistic regression + XGBoost on historical tournament games (2010–2021). Use [Elo diff, seed diff, win rate diff, point diff diff, SOS diff] as features. Holdout = 2022–2024. Report Brier Score for each model.

4. CALIBRATE: Apply isotonic regression calibration to both models on the holdout. Show calibration curves. Save calibrated probabilities.

5. BLEND: Blend logistic (40%) + XGBoost (60%) calibrated predictions. Test the blend Brier Score on holdout. If worse, try 50/50.

6. SUBMIT: Generate final_submission.csv for all 2026 matchups. Validate format. Confirm all predictions between 0.025 and 0.975 (clip extremes).

□ Iteration Tip — After Your First Submission

Once you have a score on the Kaggle leaderboard, ask your AI:

'My current Brier Score is [X]. What are the 3 most likely improvements that would give the biggest drop in score for this competition, ranked by expected impact?'

Common answers: better calibration, adding Elo, using recency-weighted features, adding conference strength.

Pick one thing at a time, re-submit, and track whether the score actually improved.

A change that helps on your holdout sometimes hurts on the real leaderboard — that's normal. Keep iterating.

One submission beats no submission every time.

Don't wait for a perfect model. Submit something early, see your score, and improve from there.

We'll be cheering for you in #march-ml-mania. □