Catch-Up Plan: From Scratch to Machine Learning

Student profile: teenager, codes in Scratch, targeting top CS programs. **Goal:** Be able to independently understand and modify Python code for Machine Learning (ML) **Budget:** 17 one-hour sessions + self-study between sessions.

Skills the student needs to be able to do Machine Learning in Python

To work with Machine Learning (ML) models in Python, the student must be comfortable with:

Skill	**Why ML needs it**
Variables, types, print	Every cell uses them
Lists and dictionaries	Data is built from dictionaries of lists
`for` loops, `if/else`	Results loop, one-hot encoding logic
Functions (`def`, `return`, args)	Understanding `model.fit()`, `model.predict()`, writing own functions
`import` and libraries	pandas, sklearn, matplotlib
pandas DataFrames	Loading CSV, selecting/dropping columns, `.head()`
One-hot encoding	`pd.get_dummies()` - turning words into numbers
X/y split concept	Features vs target, train vs test
Linear regression idea	What a "best fit line" means (math level: y = mx + b)
Model evaluation	What "error" means, `mean_absolute_error`

Phase 1: Python Foundations (Sessions 1-6)

Session 1 - "Hello Python, Goodbye Blocks"

Before session:

- Watch at least 32 minutes of Kevin Stratvert - Python for Beginners. **TIP:** Check if what is done in Windows works in macOS, if not look how to do it.
- Watch Getting Started with Jupyter Notebooks in VS Code - official Microsoft video: setting up, running cells, visualizing data (~10 min)

Notebook: `catch-up/session1-hello-python.ipynb`

Goal: The student writes and runs Python code, sees the Scratch parallels.

Time	**Activity**
0-10 min	Open Scratch side-by-side with the session notebook. Show "set myVar to 5" → `my_var = 5`.
10-25 min	Work through **Sections 1-2** of the notebook: variables, `print()`, types (`str`, `int`, `float`, `bool`) with car examples. Student runs each cell and predicts the output before running it.
25-40 min	Work through **Sections 3-5**: f-strings, changing variables, Scratch vs Python comparison table.

Time	Activity
40-55 min	**Section 6** (car listing) together, then student works on **Challenges 1-3** (dream car, dealership calculator, depreciation).
55-60 min	Recap: "In Scratch you drag, in Python you type. Same ideas, different language."

Self-study before next session: Finish **challenges 4-5** from the notebook (type detective, car comparison).

Session 2 - Lists and Loops

Notebook: `catch-up/session2-lists-and-loops.ipynb`

Goal: The student can create lists, loop through them, and use `if/else`.

Time	Activity
0-10 min	Review Session 1 challenges. Fix any confusion.
10-25 min	Work through **Sections 1-2** of the notebook: lists (indexing, `append`, `len`), `for` loops with car brands and prices. Scratch parallel: "item 1 of list" and "repeat for each". Stress that Python counts from 0.
25-40 min	Work through **Sections 3-4**: `if/elif/else` and combining loops + conditionals (categorizing cars by price).
40-55 min	**Section 5** (building a list inside a loop), then student works on **Challenges 1-3** (showroom announcer, mileage check, price tags with `zip()`).
55-60 min	Recap: "Loops + conditions = the ability to process any amount of data automatically."

Self-study:

- Complete **challenges 4-5** from the notebook (budget shopping, dealership report).
- Play levels 1-10 on CheckiO - Python (free, game-like coding challenges).

Session 3 - Dictionaries and Functions

Notebook: `catch-up/session3-dictionaries-and-functions.ipynb`

Goal: The student understands dictionaries (needed for DataFrames) and can write simple functions.

Time	Activity

Time	Activity
0-10 min	Review Session 2 challenges. Work through **Section 1**: dictionaries as labeled storage, accessing values by key, modifying and looping with `.items()`. Analogy: "a dictionary is like a Scratch list but each item has a label instead of a number."
10-25 min	**Section 2**: dictionary of lists - the exact structure that powers DataFrames. Print a manual table from a dictionary. "Next session, pandas will do this formatting for us."
25-40 min	**Section 3**: functions with `def`. Start with `honk()`, build up to parameters, then `return`. Scratch parallel: "My Blocks" / custom blocks. Practice `is_affordable()` and `total_with_tax()`.
40-55 min	**Section 4** (putting it together: car lookup system), then student works on **Challenges 1-3** (car spec sheet, depreciation function, price categorizer function).
55-60 min	Recap: "Dictionaries label data, functions package logic. Next session, pandas combines both ideas."

Self-study:

- Complete **challenges 4-5** from the notebook (dealership stats function, build your own `one_hot()` - a sneak peek at encoding!).
- Go back and watch (or re-watch) from minute 41 Kevin Stratvert - Python for Beginners.
- If *functions* are not clear enough, watch Corey Schafer - Python Functions with clear 15-min explanation of `def`, parameters, and `return`.

Session 4 - Libraries and pandas Intro

Notebook: `catch-up/session4-libraries-and-pandas.ipynb`

Goal: The student can import pandas, create a DataFrame, and explore it.

Time	Activity
0-10 min	Review Session 3 challenges. Work through **Sections 1-2**: what is a library (Scratch parallel: Extensions), `import pandas as pd`, creating a DataFrame from a dictionary. "One line replaces all our manual table printing!"
10-25 min	**Sections 3-5**: exploring a DataFrame (`.head()`, `.shape`, `.columns`, `.dtypes`, `.describe()`), selecting and dropping columns. Explain `axis=1` (columns, not rows).
25-40 min	**Sections 6-7**: sorting, filtering with boolean masks (`df[df["price"] < 20000]`), counting unique values.

Time	Activity
40-55 min	**Section 8**: reading a real CSV file (`pd.read_csv()`). Explore the project data together - how many rows? what columns? any missing values? Then student works on **Challenges 1-3**.
55-60 min	Recap: "A DataFrame is just a fancy spreadsheet that Python can manipulate. Next session: encoding text into numbers for ML."

Self-study:

- Complete **Challenges 4-5** from the notebook (real CSV exploration, X/y split preview for ML).
- Open the CSV in a spreadsheet app (Excel/Google Sheets) AND in pandas. Compare.

Session 5 - One-Hot Encoding (the Conceptual Bridge)

Goal: The student understands *why* and *how* we turn words into numbers.

Time	Activity
0-10 min	Warm-up question: "Can a calculator multiply 'Ford' times 3?" No. Computers need numbers.
10-25 min	Manual encoding on paper: draw a table with brands Ford, Toyota, BMW. Add columns `is_Ford`, `is_Toyota`, `is_BMW`. Fill in 1s and 0s together. This is one-hot encoding.
25-40 min	Walk through `some_utilities.py` line by line. The student should predict what each loop does before running it. Focus on the inner loop: "for each car, is it equal to this brand? 1 or 0."
40-50 min	Now show the pandas shortcut: `pd.get_dummies(df, columns=["brand"])`. Compare output to the manual version. "Same result, one line of code."
50-60 min	Walk through n2 notebook from start to finish. Student runs each cell.

Self-study:

- On paper, one-hot encode this list by hand: `["Red", "Blue", "Red", "Green", "Blue"]`. Write out the full table.
- Re-read `some_utilities.py` and add a comment above every line explaining what it does in your own words.

Session 6 - Train/Test Split and Putting It Together

Goal: The student understands why we split data and can prepare a dataset for ML.

Time	Activity

Time	Activity
0-10 min	Analogy: "Imagine studying for a math test. You practice with 80 problems. Then you take a test with 20 NEW problems. If you score well on problems you've never seen, you actually learned." That's train/test split.
10-20 min	Code it: `X = df_encoded.drop("price", axis=1)` and `y = df_encoded["price"]`. Explain: X = the questions (features), y = the answers (price).
20-35 min	`train_test_split(X, y, test_size=0.2, random_state=42)`. Explain each argument. Let student change `test_size` and see how the counts change. Explain `random_state` as "the shuffle seed".
35-55 min	Full walkthrough: build the n2 pipeline from scratch in a blank notebook (not copying - typing from memory with hints). Dictionary → DataFrame → get_dummies → X/y split → train_test_split.
55-60 min	"Next session: we give this data to an AI and it learns to predict prices."

Self-study:

- Recreate the full pipeline (dictionary → split) from memory in a new notebook cell. If stuck, peek at n2 and try again.
- Watch: StatQuest - Machine Learning Fundamentals (10 min, very visual).

Phase 2: Machine Learning Core (Sessions 7-11)

Session 7 - What is Linear Regression? (Math Day)

Goal: The student understands y = mx + b and how a computer finds the best line.

Time	Activity
0-15 min	On paper or whiteboard: plot 5 cars as dots - X axis = mileage, Y axis = price. Ask: "Can you draw a line that gets close to all dots?"
15-30 min	Introduce `y = mx + b`. m = slope (how much price drops per mile), b = starting price. Try two different lines and measure which one is "less wrong" by measuring distances from dots to line.
30-45 min	Do the same in Python with matplotlib: plot the car data as a scatter plot. `plt.scatter(df["mileage"], df["price"])`.
45-55 min	Key insight: "Linear Regression is an algorithm that tries millions of lines and picks the one with the smallest total error. That's it."
55-60 min	Preview: "Next time, we let Python find that line automatically."

Self-study:

- Khan Academy: Intro to Linear Regression (watch + do exercises, ~30 min).
- Plot a scatter of year vs. price from the mock data. Does it trend up or down?

Session 8 - Training Our First Model

Goal: The student trains a LinearRegression model and makes predictions.

Time	Activity
0-10 min	Quick recap: what does `train_test_split` give us? (4 things: X_train, X_test, y_train, y_test).
10-25 min	Three magic lines: `model = LinearRegression()`, `model.fit(X_train, y_train)`, `predictions = model.predict(X_test)`. Explain each: create → study → answer.
25-40 min	Run it on the mock data from n2. Print actual vs. predicted side by side. "How close was the AI?"
40-50 min	Introduce `mean_absolute_error(y_test, predictions)`. "On average, the AI was off by $X." Is that good or bad? Discuss.

Time	Activity
50-60 min	Custom prediction: `model.predict([[2019, 40000, 0, 0, 0, 1]])`. "We just asked the AI: what would a 2019 Toyota with 40k miles cost?" Change the values and see what happens.

Self-study:

- Predict 3 different cars by changing the custom_car values. Write down your predictions and whether they make sense.
- Re-read the n3 notebook. How much of it do you now understand?

Session 9 - Understanding n3 End to End

Goal: The student can walk through `n3_price_engine.ipynb` and explain every line.

Time	Activity
0-5 min	Open n3. "Today you're the teacher. Explain each cell to me."
5-30 min	Student explains cell by cell. Instructor only intervenes to correct misconceptions. If student gets stuck, give a hint, don't give the answer.
30-45 min	Notice n3 reads from `usedcarprices_sujayr_train.csv` - a real dataset with more columns. Explore it together: what columns are there? Which are numbers, which are text?
45-55 min	Discussion: "This dataset has columns like Engine, Power, Fuel_Type. Which ones would you use as features? Which need encoding?"
55-60 min	Identify gaps in n3 (missing preprocessing, missing encoding steps between data load and model training) and discuss what would be needed to make it run end-to-end.

Self-study:

- Write a paragraph (plain English, not code) describing what n3 does, step by step, as if explaining to a friend.
- List 3 columns from the CSV that you think are most important for predicting price and why.

Session 10 - Fixing and Extending n3

Goal: The student adds missing preprocessing steps to make n3 run end-to-end on the real CSV.

Time	Activity
0-10 min	Review: n3 loads data but jumps straight to model training without encoding or feature selection. Let's fix that.

Time	Activity
10-25 min	Together, select numeric columns (Year, Kilometers_Driven, Seats) and encode Fuel_Type and Transmission using get_dummies. Handle missing values with `df.dropna()` (simplest approach).
25-40 min	Build the full pipeline: load → clean → encode → X/y split → train/test split → train → predict → evaluate. Student types, instructor guides.
40-55 min	Run it. Check the error. "Is $X average error good for a car price?" Discuss what "good" means in context.
55-60 min	Celebrate: "You just built a working AI that predicts car prices from real data."

Self-study:

- Try adding or removing features (columns) and see how the error changes. Write down what you tried and the error for each.
- Start thinking: what other data would make this model better?

Session 11 - Review and Confidence Check

Goal: Solidify understanding. Student should be able to build a mini ML pipeline without help.

Time	Activity
0-5 min	"Today is your final exam on this phase. I'll give you a small dataset and you build the whole thing."
5-40 min	Provide a simple new dataset (e.g., house prices with 3-4 columns, or make one up with 15 rows). Student builds: DataFrame → encoding → split → train → predict → evaluate. Instructor stays silent unless student is stuck for >3 minutes.
40-50 min	Review together. What went well? What was tricky?
50-60 min	Preview of Phase 3: "Now that you can build a model, let's learn to build a BETTER model."

Self-study:

- Find a small, clean CSV dataset on Kaggle (suggestions: tips dataset, iris dataset). Try to build a prediction pipeline on your own.

Phase 3: Going Deeper (Sessions 12-17)

These sessions extend beyond n3 into the territory needed for n4 and for a strong college application portfolio.

Session 12 - Data Cleaning in the Wild

Working with `ussalescars_juanmerino_train.zip`: missing values, outliers, data types. Learn `df.isnull().sum()`, `df.dropna()`, `df.fillna()`, filtering with boolean masks (`df[df["price"] < 100000]`).

Session 13 - Feature Engineering

Creating new features from existing ones: car age from year (`2024 - df["Year"]`), price per mile, grouping rare categories into "Other". Understanding which features help and which add noise.

Session 14 - Beyond Linear Regression

Try `DecisionTreeRegressor` and `RandomForestRegressor` from sklearn. Same `.fit()` / `.predict()` interface. Compare errors across models. Discuss: why do some models work better?

Session 15 - Visualizing Results

Using matplotlib to tell the story: scatter plots of actual vs. predicted, bar charts of feature importance, histograms of error distribution. A good chart is worth a thousand print statements.

Session 16 - The Full Pipeline

Build a notebook for ML from scratch: load the real dataset, clean it, engineer features, encode, split, train multiple models, compare, visualize. This is the student's capstone for this project.

Session 17 - Portfolio and Presentation

Package the work: clean up notebooks, add markdown explanations, create a summary of findings. Prepare a 5-minute explanation of the project suitable for a college interview. Practice: "I built a machine learning model that predicts used car prices with $X average error. Here's how it works..."

Self-Study Resources (Reference List)

Setup and Tools (Watch First)

- Kevin Stratvert - Python for Beginners - full beginner tutorial using VSCode, covers install + basics.
- VS Code Jupyter Notebooks docs - reference for shortcuts, debugging, and features

Python Basics

- CheckiO - game-like Python challenges (do Islands of Knowledge first)
- Automate the Boring Stuff, Chapters 1-4 - free online textbook, practical examples
- Corey Schafer - Python Functions - clear 15-min explainer on `def`, parameters, and `return`

Data and pandas

- Kaggle - Intro to Pandas - free micro-course with exercises (~4 hours)
- 10 Minutes to Pandas (official docs) - good as a reference after the Kaggle course

Machine Learning Concepts

- StatQuest with Josh Starmer - visual, intuitive explanations of ML concepts (watch Linear Regression, Train/Test, Bias/Variance)
- Khan Academy - Linear Regression - the math foundation, with exercises
- Google's ML Crash Course - First 3 modules - slightly more advanced, good stretch goal

Math Prerequisites

- Khan Academy - Statistics and Probability - mean, median, standard deviation (what `df.describe()` shows)
- 3Blue1Brown - Essence of Linear Algebra - not required but impressive for college prep and helps understand what the model is really doing