# The End-to-End Machine Learning Lifecycle

A structural map of a "Toy Project" pipeline—connecting raw data to a user-facing application without getting lost in mathematical complexity.


Raw Data → Preprocessing → Training → Evaluation → Pickle File → Web Application

Based on 100 Days of Machine Learning | Day 13: End to End Toy Project

Based on 100 Days of Machine Learning | Day 13: End to End Toy Project

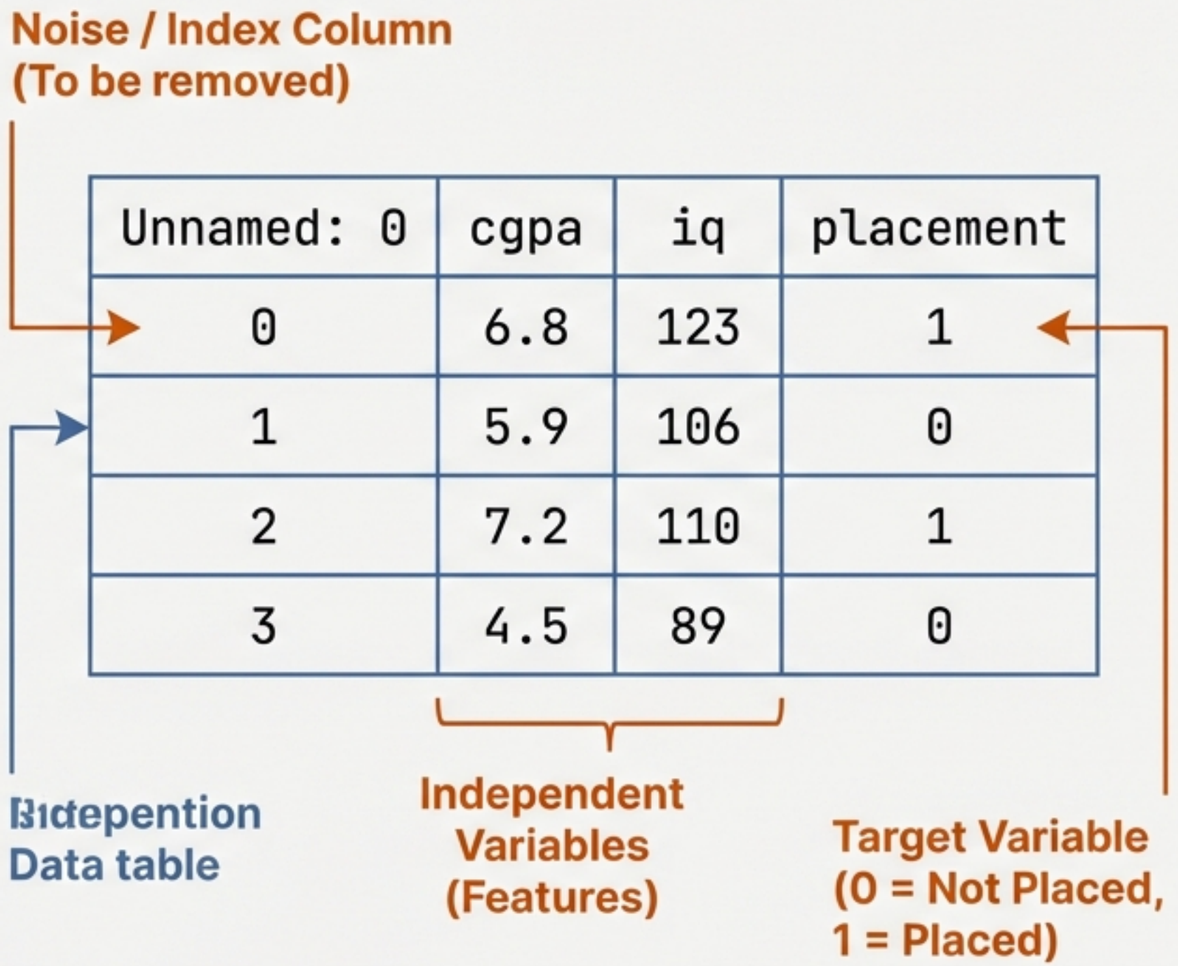
# The Raw Materials: Defining the Problem

**The Mission**
Create a predictive model to determine if a student gets placed based on academic metrics.

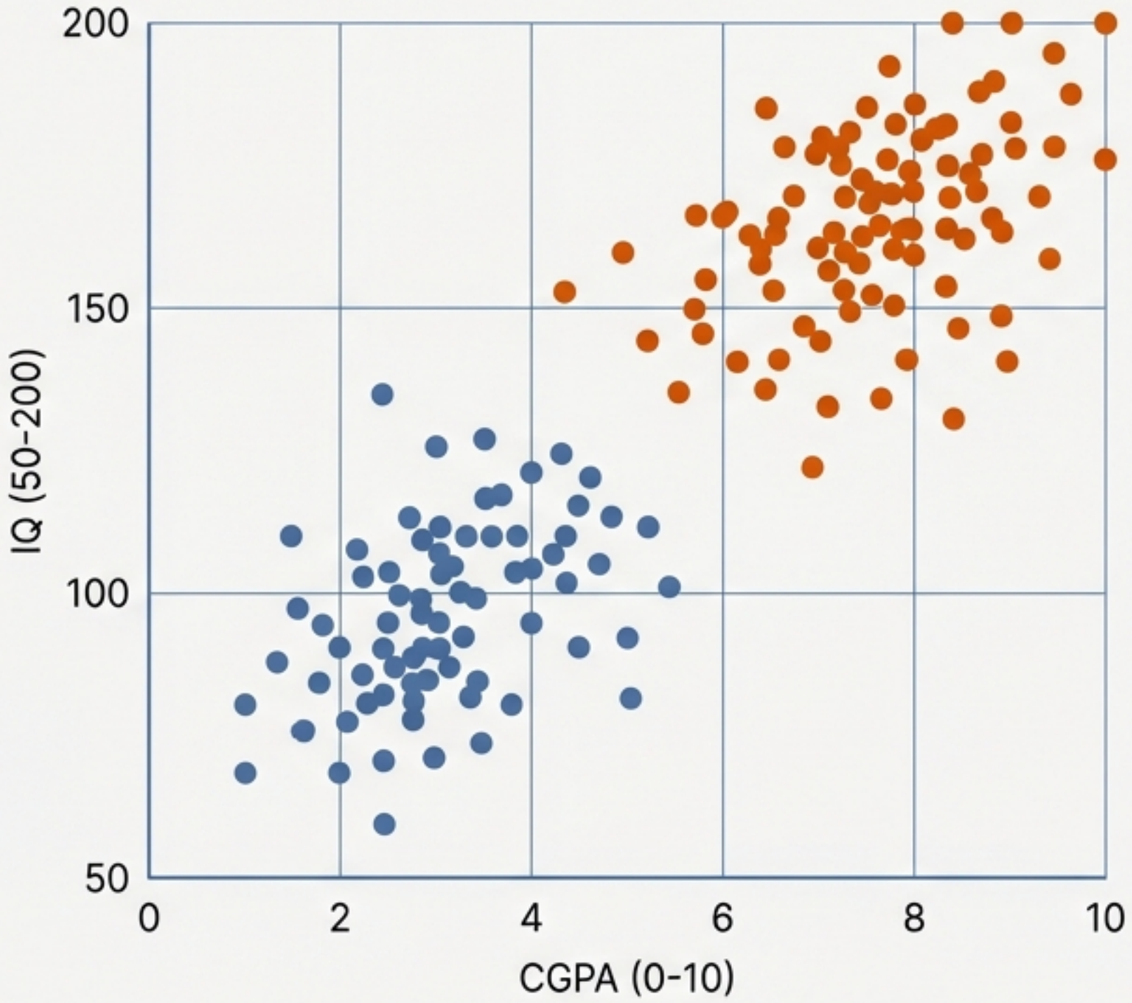
**Problem Type:**
Binary Classification

**Dataset:**
'placement.csv' (Toy Dataset)

**Scale:**
100 Rows, 4 Columns



# Visualising Hidden Patterns



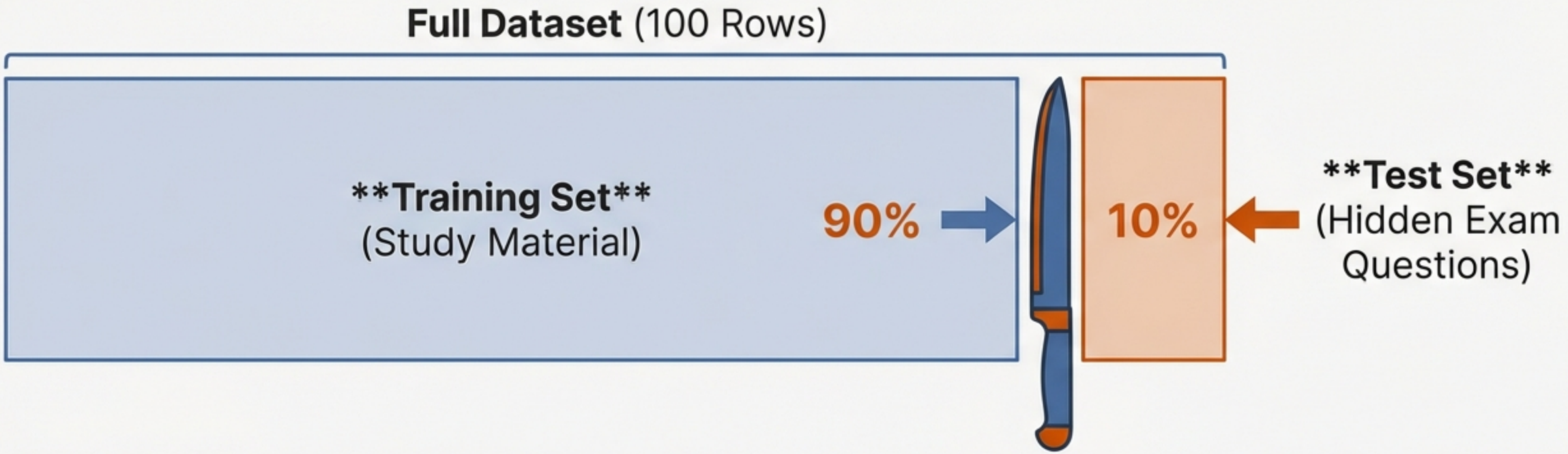

## Exploratory Data Analysis (EDA)

Before modelling, we plot the raw data to understand the relationship between features.

**Key Insight:**
The data is **Linearly Separable**. Orange dots (**Placed**) and Blue dots (**Not Placed**) cluster in distinct regions that can be divided by a straight line.

**Implication:**
We do not need complex non-linear algorithms. A simple linear classifier will suffice.

# The Study Materialteriaf vs The Final Exam




## Why Split?

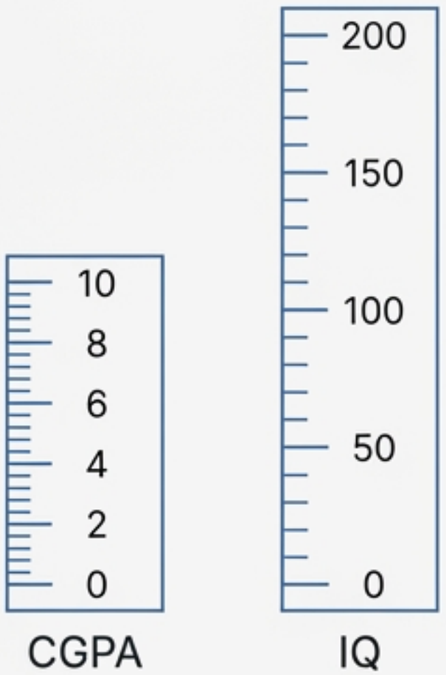
To evaluate the model fairly, we hide a portion of the data during the learning phase. This prevents the model from just memorizing answers.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
# 90 rows for training, 10 rows for testing
```

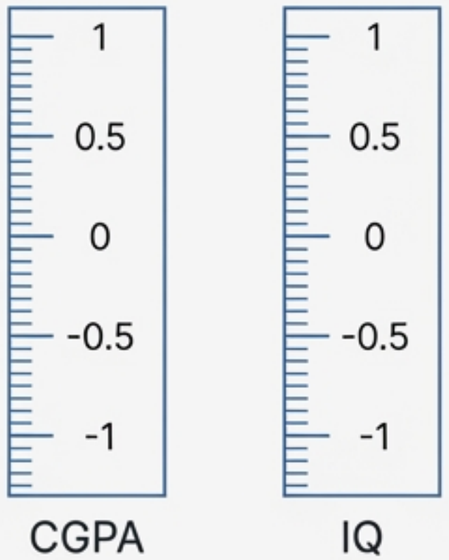
# Levelling the Playing Field

## Original Scale




Input features have vastly different magnitudes. Large numbers (IQ) can unfairly dominate mathematical distance calculations.

## After Scaling




StandardScaler transforms all features to a uniform range centered around 0.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

# Selecting the Architect: Logistic Regression

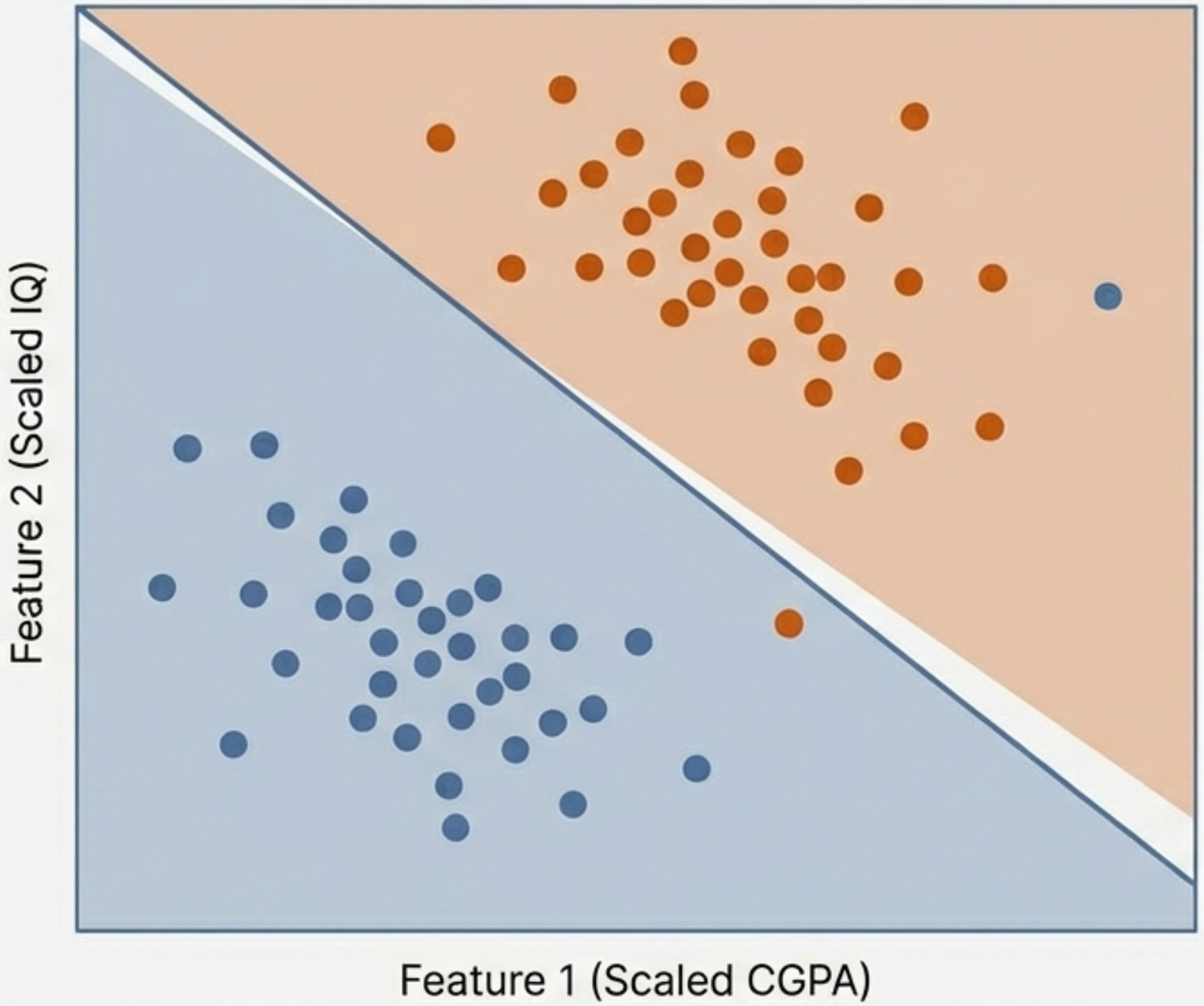
Because the data is linearly separable, we choose a linear algorithm. The 'Training' phase is where the model mathematically finds the best line to separate the classes.


```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression()

# The Learning Phase
clf.fit(X_train_scaled, y_train)
```

# Evaluation and The Decision Boundary




## The Moment of Truth

We feed the hidden Test Set (10 students) into the model.

## Metric: Accuracy Score

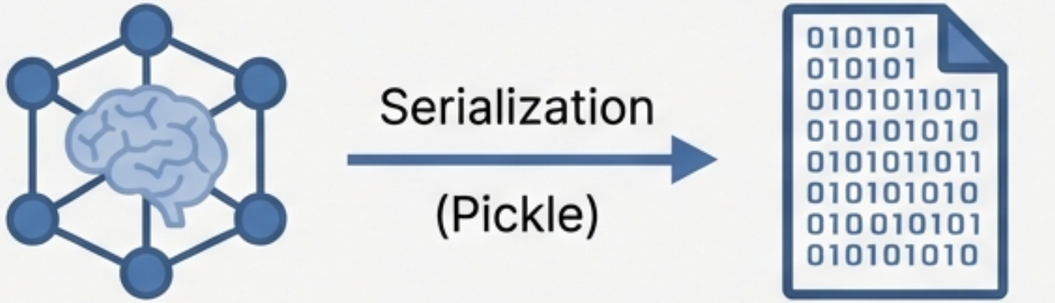
**90%**

(9 out of 10 predictions matched reality)

```
y_pred = clf.predict(X_test_scaled)
accuracy_score(y_test, y_pred)
```

# Packaging the Model for Export

The trained model currently lives only in our coding environment's memory. To move it to a web server, we must 'pickle' (serialize) it into a file.



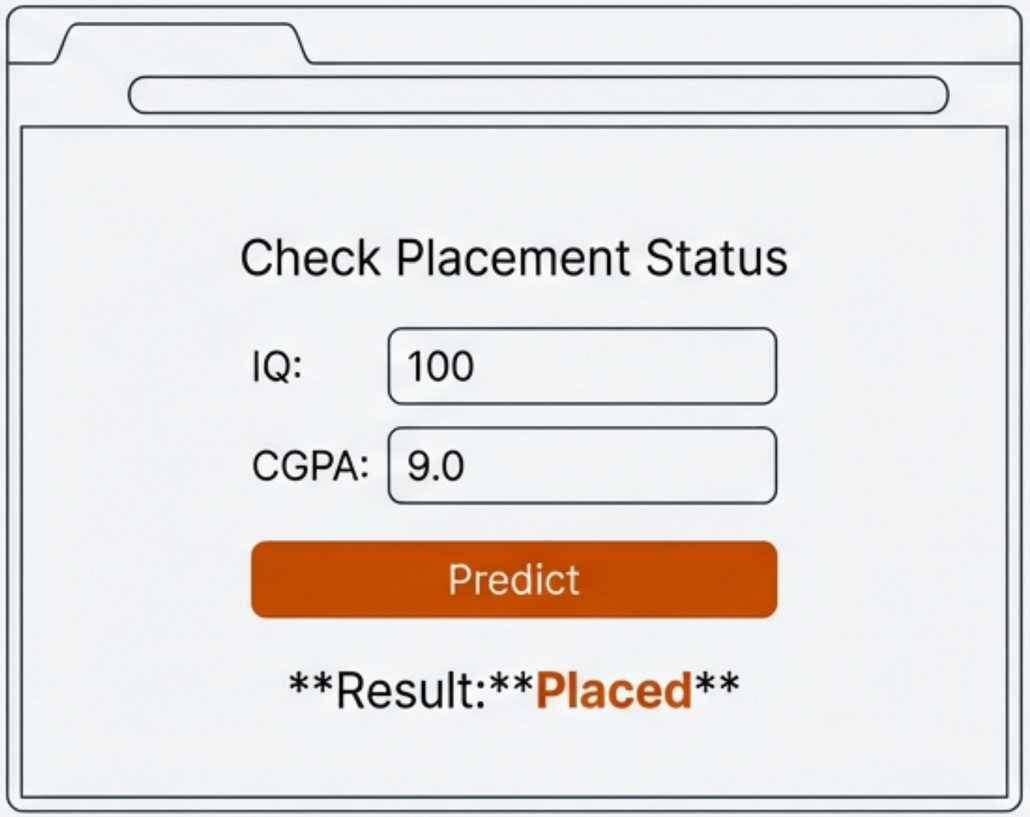

```
import pickle

pickle.dump(clf, open('model.pkl', 'wb'))
```

Think of this like freezing a cooked meal. The file contains the mathematical rules, ready to be 'reheated' (loaded) on a different computer.

# The User Interface Integration

## Frontend




## Backend Workflow

**User Input:** IQ=100, CGPA=9.0

**Server:** Loads `model.pkl`

**Computation:** `model.predict([[100, 9.0]])`

**Output:** Returns **"1"** (**Placed**) to Frontend

# Deployment and The Reality Gap

## Deployment Options

Heroku

aws

Google Cloud

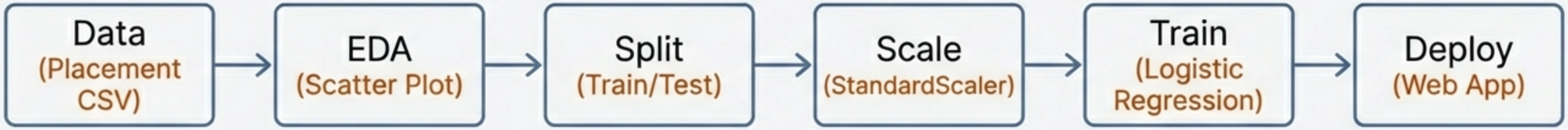
Hosting the web app so it is accessible via a public URL.

## Toy Project vs. Real World

| Toy Project | Real World |
| --- | --- |
| Clean, tiny dataset (100 rows). | Massive, messy datasets (`Millions` of rows). |
| Instant training time. | Complex cleaning & feature engineering. |
| Simple linear logic. | Slow training (Hours/Days). |
| No missing values. | Rigorous model selection & tuning. |

Despite the complexity difference, the **pipeline architecture** remains identical.

# Summary & Next Steps




**Recap:**

We successfully transformed raw data into a functional product. This workflow serves as the skeleton for every Machine Learning project, regardless of scale.

**Coming Up:**

Deep dives into each stage: Feature Selection, Advanced Scaling, Model Selection, and Automated Pipelines.