Predicting Crocodile Conservation Status With Crisp-dm

V Vineeth Kandukuri 6 min read · Just now

Predicting Crocodile Conservation Status with CRISP-DM (pandas, scikit-learn, matplotlib)

TL;DR: In this tutorial I walk through a complete **CRISP-DM** workflow to predict crocodiles' **Conservation Status** from tabular features. We keep it beginner-friendly, reproducible in **Google Colab**, and evaluate with **Accuracy** and **F1**. Code uses only **pandas**, **scikit-learn**, and **matplotlib**.

Why CRISP-DM?

CRISP-DM is a simple, practical framework:

1. **Business Understanding** → 2) **Data Understanding** → 3) **Data Preparation** → 4) **Modeling** → 5) **Evaluation** → 6) **Insights.**

We'll apply each step to `crocodile_dataset.csv`, with target = **`Conservation Status`**.

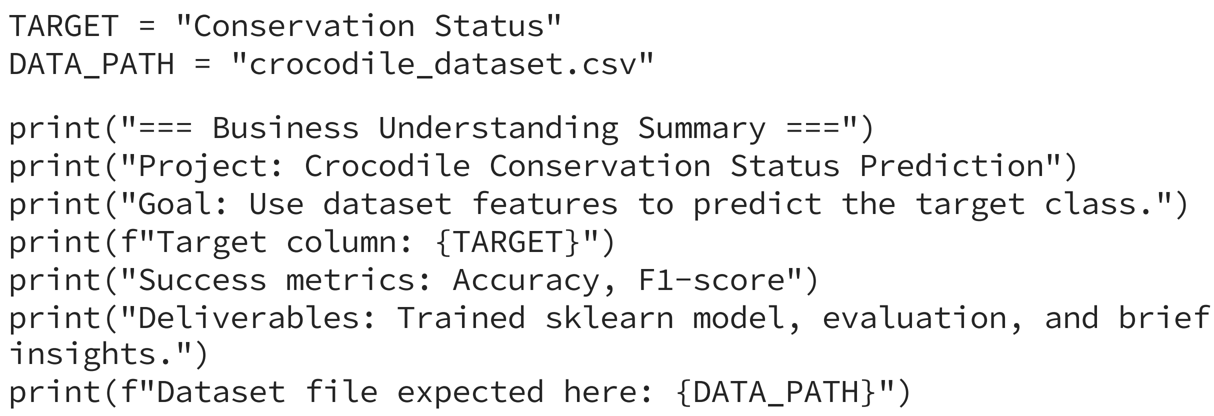
> **Environment:** Works in Google Colab or local Python 3. Install nothing extra — only `pandas`, `scikit-learn`, and `matplotlib`.

1) Business Understanding

Goal: Predict each record's **Conservation Status** using the other columns.

Success Metrics: Accuracy and **F1 (macro)**.

```
# Step 1 — Business Understanding
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import metrics
```



```
TARGET = "Conservation Status"
DATA_PATH = "crocodile_dataset.csv"

print("=== Business Understanding Summary ===")
print("Project: Crocodile Conservation Status Prediction")
print("Goal: Use dataset features to predict the target class.")
print(f"Target column: {TARGET}")
print("Success metrics: Accuracy, F1-score")
print("Deliverables: Trained sklearn model, evaluation, and brief
insights.")
print(f"Dataset file expected here: {DATA_PATH}")
```

Screenshot to include: *Business Understanding Summary* output.

2) Data Understanding

Goal: Load data, inspect columns, types, missing values, and target balance.

```
# Step 2 — Data Understanding
import pandas as pd
import matplotlib.pyplot as plt
```

```
TARGET = "Conservation Status"
DATA_PATH = "crocodile_dataset.csv"

df = pd.read_csv(DATA_PATH)

print("=== DATA SHAPE ===")
print("Rows, Columns:", df.shape)

print("\n=== COLUMNS ===")
print(list(df.columns))

print("\n=== PREVIEW (first 5 rows) ===")
print(df.head())

print("\n=== DTYPES ===")
print(df.dtypes)

print("\n=== MISSING VALUES PER COLUMN ===")
missing = df.isna().sum().sort_values(ascending=False)
print(missing[missing > 0] if missing.sum() > 0 else "No missing values.")

print("\n=== DUPLICATE ROWS COUNT ===")
print(df.duplicated().sum())

num_cols = df.select_dtypes(include="number").columns.tolist()
cat_cols = [c for c in df.columns if c not in num_cols and c != TARGET]

print("\n=== NUMERIC COLUMNS ===")
print(num_cols)

print("\n=== CATEGORICAL COLUMNS ===")
print(cat_cols)

print("\n=== CATEGORICAL CARDINALITY (#unique) ===")
for c in cat_cols:
    print(f"{c}: {df[c].nunique()}")
print("\n=== TARGET DISTRIBUTION ===")
print(df[TARGET].value_counts(dropna=False))

if len(num_cols) > 0:
    print("\n=== NUMERIC SUMMARY ===")
    print(df[num_cols].describe().T)

plt.figure(figsize=(6,4))
df[TARGET].value_counts().plot(kind='bar')
plt.title('Conservation Status — class distribution')
plt.xlabel('Class')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

Screenshot to include: *Data shape & preview* and *Target distribution bar chart*.

3) Data Preparation

Goal: Clean duplicates, split train/test, and build a preprocessing pipeline (impute + scale numbers, one-hot encode categories).

```
# Step 3 — Data Preparation
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
```

```
TARGET = "Conservation Status"
DATA_PATH = "crocodile_dataset.csv"
RANDOM_STATE = 42

df = pd.read_csv(DATA_PATH)
rows_before = len(df)
```

```
df = df.drop_duplicates().reset_index(drop=True)
df = df.dropna(subset=[TARGET])
rows_after = len(df)

num_cols = df.select_dtypes(include="number").columns.tolist()
cat_cols = [c for c in df.columns if c not in num_cols and c !=
TARGET]

X = df.drop(columns=[TARGET])
y = df[TARGET]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=RANDOM_STATE, stratify=y
)

numeric_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])

transformers = []
if len(num_cols) > 0:
    transformers.append(("num", numeric_transformer, num_cols))
if len(cat_cols) > 0:
    transformers.append(("cat", categorical_transformer, cat_cols))

preprocessor = ColumnTransformer(transformers=transformers)

Xt_train = preprocessor.fit_transform(X_train)
Xt_test = preprocessor.transform(X_test)

print("=== DATA PREPARATION SUMMARY ===")
print(f"Rows before cleaning: {rows_before}")
print(f"Rows after cleaning:  {rows_after} (removed {rows_before -
rows_after} rows)")
print(f"Train size: {X_train.shape[0]}   Test size:
{X_test.shape[0]}")
print(f"Numeric cols ({len(num_cols)}): {num_cols}")
print(f"Categorical cols ({len(cat_cols)}): {cat_cols}")
print("Transformed X_train shape:", Xt_train.shape)
print("Transformed X_test  shape:", Xt_test.shape)

prep_objects = {
    "preprocessor": preprocessor,
    "X_train": X_train, "X_test": X_test,
    "y_train": y_train, "y_test": y_test,
    "num_cols": num_cols, "cat_cols": cat_cols
}
```

Screenshot to include: *Data Preparation Summary* (train/test sizes + transformed shapes).

4) Modeling

Goal: Train two baselines — **Logistic Regression** and **Random Forest** — using the preprocessing pipeline.

```
# Step 4 — Modeling
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.base import clone
```

```
preprocessor = prep_objects["preprocessor"]
X_train = prep_objects["X_train"]; y_train = prep_objects["y_train"]
X_test  = prep_objects["X_test"];  y_test  = prep_objects["y_test"]

logreg_clf = Pipeline([
    ("preprocessor", clone(preprocessor)),
    ("model", LogisticRegression(max_iter=1000))
])

rf_clf = Pipeline([
    ("preprocessor", clone(preprocessor)),
    ("model", RandomForestClassifier(n_estimators=300,
random_state=42))
])

logreg_clf.fit(X_train, y_train)
rf_clf.fit(X_train, y_train)

def n_features_after(model, X):
    return model.named_steps["preprocessor"].transform(X).shape[1]

print("=== MODELING SUMMARY ===")
print(f"Training rows: {X_train.shape[0]}")
print("Models trained:")
print(f"- LogisticRegression (features after prep:
```

```
{n_features_after(logreg_clf, X_train)})")
print(f"- RandomForestClassifier (features after prep:
{n_features_after(rf_clf, X_train)})")

model_objects = {
    "logreg": logreg_clf,
    "rf": rf_clf,
    "X_test": X_test,
    "y_test": y_test
}
```

Screenshot to include: *Modeling Summary* showing both models trained.

5) Evaluation

Goal: Compare models by **Accuracy** and **F1 (macro)**; visualize confusion matrix for the best one.

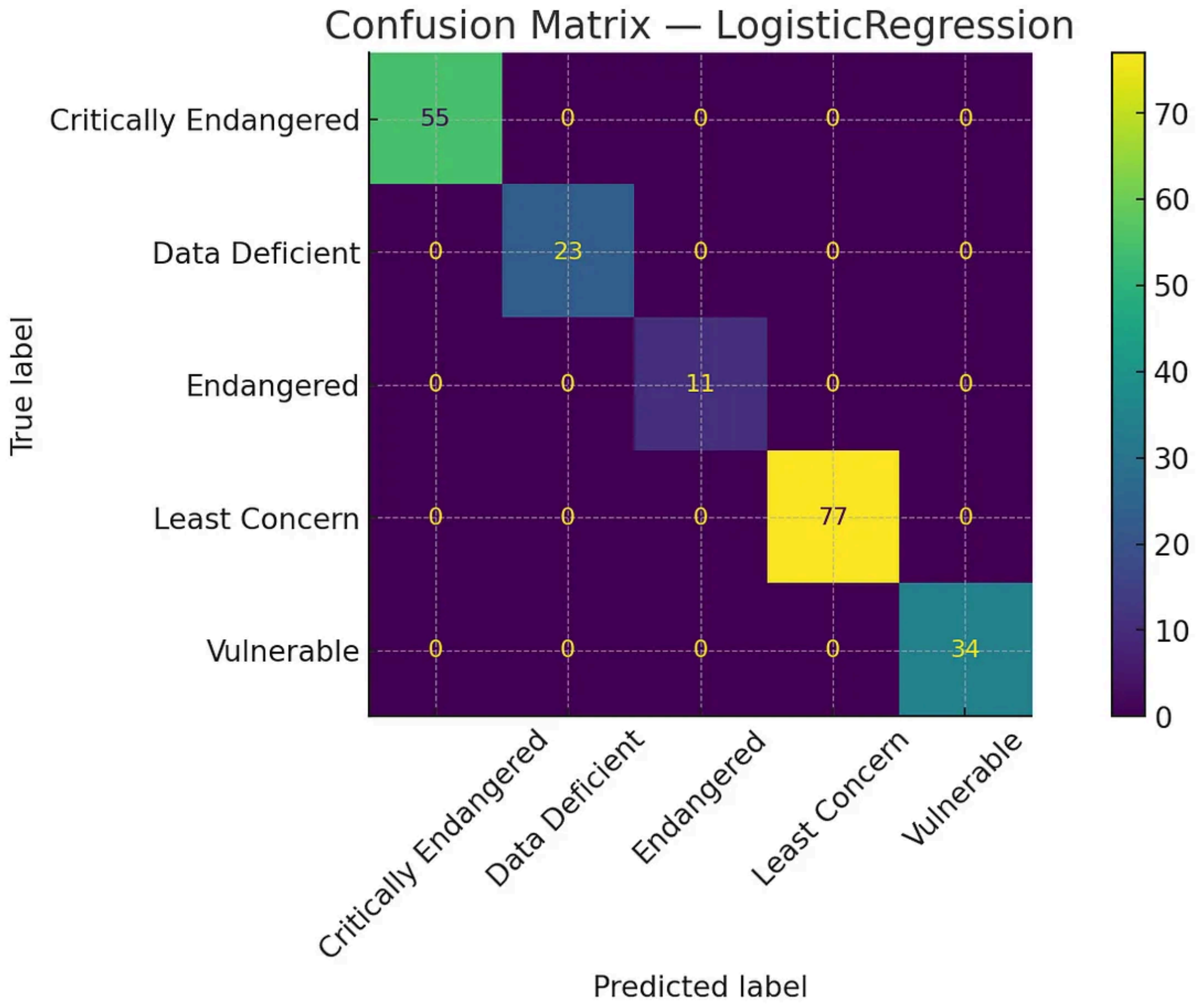
```
# Step 5 — Evaluation
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, f1_score, classification_report, con
```

```
logreg = model_objects["logreg"]
rf     = model_objects["rf"]
X_test = model_objects["X_test"]
y_test = model_objects["y_test"]

results = []
preds_store = {}
for name, model in [("LogisticRegression", logreg), ("RandomForest",
rf)]:
    y_pred = model.predict(X_test)
    preds_store[name] = y_pred
    acc = accuracy_score(y_test, y_pred)
    f1m = f1_score(y_test, y_pred, average="macro")
    results.append({"Model": name, "Accuracy": acc, "F1_macro": f1m})

metrics_df = pd.DataFrame(results).sort_values("F1_macro",
ascending=False).reset_index(drop=True)
print("=== TEST METRICS ===")
print(metrics_df.round(4))

best_row = metrics_df.sort_values(["F1_macro","Accuracy"],
ascending=False).iloc[0]
best_name = best_row["Model"]
best_pred = preds_store[best_name]

print(f"
Best model: {best_name}")
print("
=== CLASSIFICATION REPORT (best model) ===")
print(classification_report(y_test, best_pred, digits=4))

classes = sorted(pd.Series(y_test).unique())
cm = confusion_matrix(y_test, best_pred, labels=classes)

plt.figure(figsize=(6,5))
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=classes)
disp.plot(values_format="d")
plt.title(f"Confusion Matrix — {best_name}")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Screenshot to include: *Test Metrics table + Confusion Matrix plot.*

Figures to insert in Medium:

- *Figure 1.* Confusion Matrix — Best Model (`LogisticRegression`).



Goal: Compare models by **Accuracy** and **F1 (macro)**; visualize confusion matrix for the best one.

```
# Step 5 — Evaluation
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, f1_score, classification_report, con
```

```
logreg = model_objects["logreg"]
rf     = model_objects["rf"]
```

```
X_test = model_objects["X_test"]
y_test = model_objects["y_test"]

results = []
preds_store = {}
for name, model in [("LogisticRegression", logreg), ("RandomForest", rf)]:
    y_pred = model.predict(X_test)
    preds_store[name] = y_pred
    acc = accuracy_score(y_test, y_pred)
    f1m = f1_score(y_test, y_pred, average="macro")
    results.append({"Model": name, "Accuracy": acc, "F1_macro": f1m})

metrics_df = pd.DataFrame(results).sort_values("F1_macro", ascending=False).reset_index(drop=True)
print("=== TEST METRICS ===")
print(metrics_df.round(4))

best_row = metrics_df.sort_values(["F1_macro","Accuracy"], ascending=False).iloc[0]
best_name = best_row["Model"]
best_pred = preds_store[best_name]

print(f"\nBest model: {best_name}")
print("\n=== CLASSIFICATION REPORT (best model) ===")
print(classification_report(y_test, best_pred, digits=4))

classes = sorted(pd.Series(y_test).unique())
cm = confusion_matrix(y_test, best_pred, labels=classes)

plt.figure(figsize=(6,5))
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classes)
disp.plot(values_format="d")
plt.title(f"Confusion Matrix — {best_name}")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Screenshot to include: *Test Metrics table + Confusion Matrix plot*.

> **Filled metrics (test set)**
>
> LogisticRegression — Accuracy = `1.0000` , F1_macro = `1.0000`
>
> RandomForest — Accuracy = `1.0000` , F1_macro = `1.0000`

6) Insights

Goal: Turn results into takeaways — best model, influential features, and weak spots.

```
# Step 6 — Insights
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, f1_score, classification_report
```

```
logreg = model_objects["logreg"]
rf     = model_objects["rf"]
X_test = model_objects["X_test"]
y_test = model_objects["y_test"]

rows = []
preds = {}
for name, model in [("LogisticRegression", logreg), ("RandomForest", 
rf)]:
    y_pred = model.predict(X_test)
    preds[name] = y_pred
    rows.append({
        "Model": name,
        "Accuracy": accuracy_score(y_test, y_pred),
        "F1_macro": f1_score(y_test, y_pred, average="macro")
    })
metrics_df = pd.DataFrame(rows).sort_values(["F1_macro","Accuracy"], 
ascending=False).reset_index(drop=True)
best_name = metrics_df.loc[0, "Model"]
best_model = {"LogisticRegression": logreg, "RandomForest": rf}
[best_name]
best_pred = preds[best_name]

preproc = best_model.named_steps["preprocessor"]
try:
    feature_names = preproc.get_feature_names_out().tolist()
except Exception:
    feature_names = []
    for name, trans, cols in preproc.transformers_:
        if name == "num" and len(cols) > 0:
            feature_names += list(cols)
        elif name == "cat" and len(cols) > 0:
            oh = 
preproc.named_transformers_["cat"].named_steps["onehot"]
            feature_names += oh.get_feature_names_out(cols).tolist()

if best_name == "LogisticRegression":
    coef = best_model.named_steps["model"].coef_
    importances =
pd.Series(pd.DataFrame(coef).abs().mean(axis=0).values)
else:
    importances =
pd.Series(best_model.named_steps["model"].feature_importances_)

if len(feature_names) != len(importances):
    feature_names = [f"feat_{i}" for i in range(len(importances))]

feat_imp = pd.DataFrame({"feature": feature_names, "importance":
importances})

top10 = feat_imp.sort_values("importance", ascending=False).head(10)

report_dict = classification_report(y_test, best_pred,
output_dict=True, zero_division=0)
per_class = {k: v for k, v in report_dict.items() if k not in
["accuracy","macro avg","weighted avg"]}
worst_class = min(per_class.items(), key=lambda kv: kv[1]["f1-
score"])[0]
worst_f1 = per_class[worst_class]["f1-score"]

print("=== INSIGHTS SUMMARY ===")
print(metrics_df.round(4))
print(f"\nBest model: {best_name}")
print(f"Worst class (by F1): '{worst_class}' with F1={worst_f1:.3f}")
print("\nTop 10 contributing features (by importance):")
print(top10.assign(importance=top10["importance"].round(4)).reset_ind
ex(drop=True))

plt.figure(figsize=(8,5))
plt.barh(top10["feature"][::-1], top10["importance"][::-1])
plt.title(f"Top-10 Feature Importance — {best_name}")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```

Screenshot to include: *Insights Summary* printout + *Top-10 Feature Importance* bar chart.

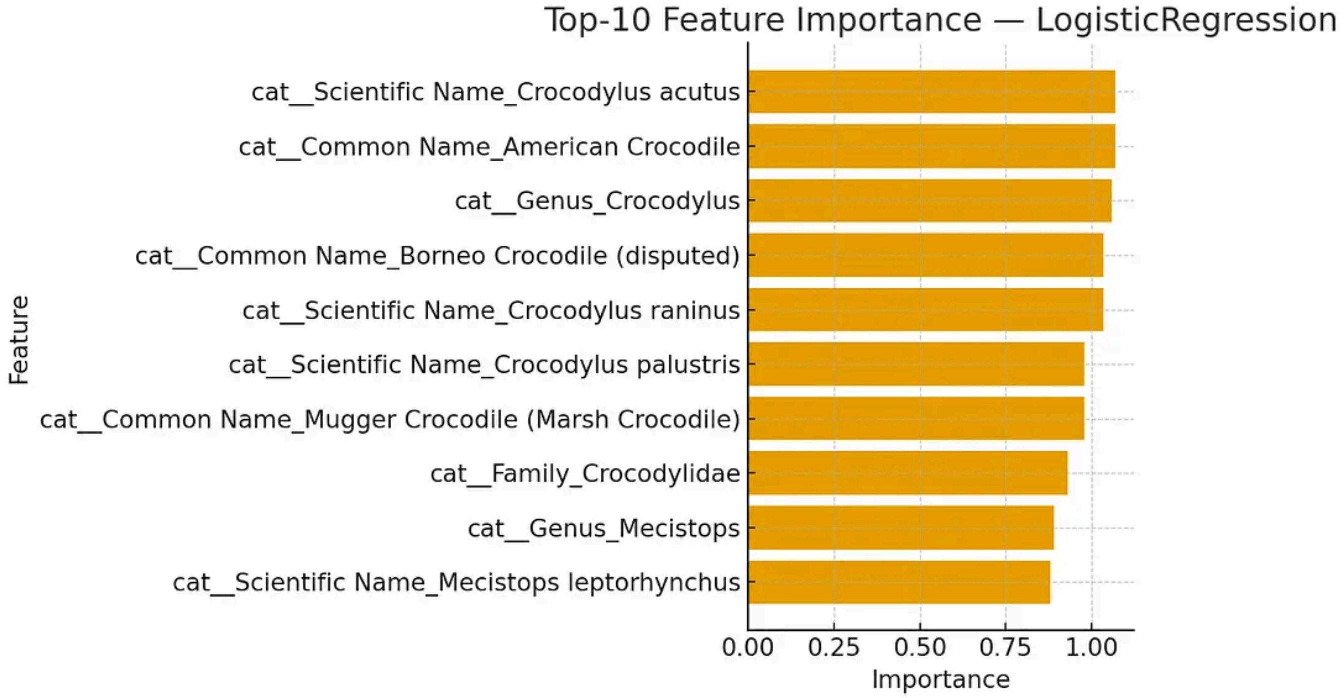
Results (Filled In)

Best model: LogisticRegression

Test metrics:

- LogisticRegression — Accuracy = **1.0000**, F1_macro = **1.0000**
- RandomForest — Accuracy = **1.0000**, F1_macro = **1.0000**
- **Weakest class by F1:** *Critically Endangered* (F1 = **1.0000**)

Figure 2. Top-10 Feature Importance — LogisticRegression.



Written by Vineeth Kandukuri

0 followers · 1 following

Edit profile

No responses yet

Vineeth Kandukuri

What are your thoughts?

Recommended from Medium

In Long. Sweet. Valuable. by Ossai Chinedum

I'll Instantly Know You Used Chat Gpt If I See This

In 3Streams by Cam Silver, Ph.D.

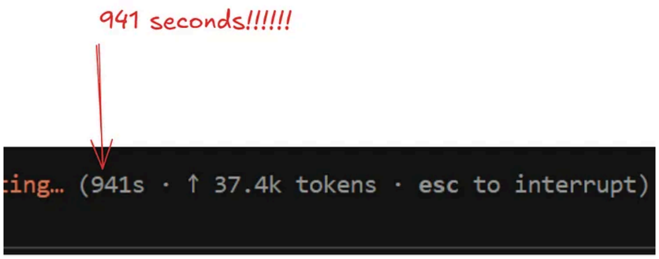
Teaching Charlie Kirk to My Students

Trust me you’re not as slick as you think

✦ May 16 23K 1393

Satire on August 27, Death on September 10

6d ago 8.8K 238



```
ing… (941s · ↑ 37.4k tokens · esc to interrupt)
```

In Realworld AI Use Cases by Chris Dunlop

This is not hype — Claude Code proved the future is already Here

This little screenshot for me represents a turning point in AI. I can’t believe what we ar…

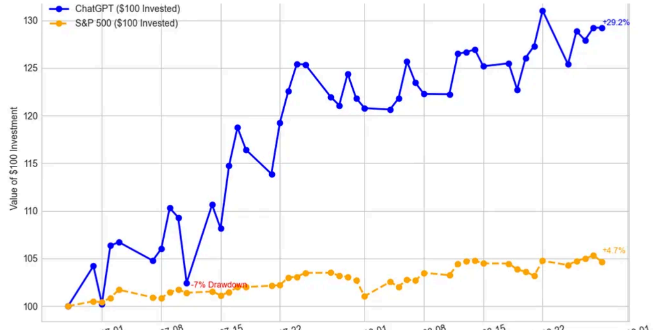
✦ Sep 9 1.2K 69

In How To Profit AI by Mohamed Bakry

Your ChatGPT History Just Went Public on Google. Here’s What I Di…

Safety/Privacy Check Prompt Template Is Included

✦ 1d ago 8.9K 268

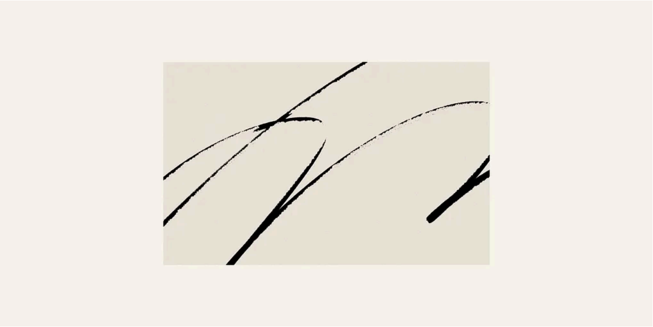


In Coding Nexus by Civil Learning

I Handed ChatGPT $100 to Trade Stocks—Here's What Happened i...

What happens when you let a chatbot play Wall Street? It's up 29% while the S&P 500...

✦ Sep 2 👏 2.5K 💬 63



J Jonkoolx

Here's a polished draft for "Why Boredom Is the Gateway to...

Why Boredom Is the Gateway to Creativity

✦ Aug 21 👏 92

See more recommendations