**Essence of Training Linear Regression Models**

Over here, I have trained a **Linear Regression** model on the BOSTON dataset with robust preprocessing (median imputation for numerics, most-frequent for categoricals, scaling, and one-hot encoding). I also saved the pretrained pipeline and showed predictions right inside the notebook context.

**What’s already completed (and available to download)**

* **Pretrained Linear Regression pipeline**: boston\_linear\_regression.pkl
* **Feature metadata** (for UI/validation): linear\_feature\_meta.json

During training (80/20 split), the model achieved:

* **RMSE** ≈ 4.9190
* **R²** ≈ 0.6701

You’ll also see two tables in your workspace:

* **Linear Regression — Test Metrics**
* **Sample Predictions (first 10 of test set)**

**A. Notebook cells (copy-paste into your Jupyter notebook)**

**1) Train and Save Linear Regression (pretrained .pkl)**

import os, json

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

import joblib

# 1) Load

df = pd.read\_csv("data.csv").replace([np.inf, -np.inf], np.nan)

# 2) Target (MEDV typical for Boston)

target = next((c for c in df.columns if c.strip().lower()=="medv"), df.columns[-1])

X = df.drop(columns=[target])

y = df[target]

# 3) Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42

)

# 4) Preprocessing

num\_feats = X.select\_dtypes(include=[np.number]).columns.tolist()

cat\_feats = [c for c in X.columns if c not in num\_feats]

numeric\_tf = Pipeline([

("imputer", SimpleImputer(strategy="median")),

("scaler", StandardScaler())

])

categorical\_tf = Pipeline([

("imputer", SimpleImputer(strategy="most\_frequent")),

("onehot", OneHotEncoder(handle\_unknown="ignore"))

])

preprocessor = ColumnTransformer([

("num", numeric\_tf, num\_feats),

("cat", categorical\_tf, cat\_feats),

])

# 5) Linear Regression pipeline

linreg\_pipe = Pipeline([

("prep", preprocessor),

("model", LinearRegression())

])

linreg\_pipe.fit(X\_train, y\_train)

# 6) Evaluate

y\_pred = linreg\_pipe.predict(X\_test)

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

r2 = r2\_score(y\_test, y\_pred)

print(f"Test RMSE: {rmse:.4f}")

print(f"Test R²: {r2:.4f}")

# 7) Save pretrained model + meta

joblib.dump(linreg\_pipe, "boston\_linear\_regression.pkl")

with open("linear\_feature\_meta.json", "w") as f:

json.dump({

"all\_features": X.columns.tolist(),

"numeric\_features": num\_feats,

"categorical\_features": cat\_feats,

"target": target

}, f, indent=2)

**2) Make Predictions inside the notebook**

**a) Predict on the test split (show a few rows)**

preview = X\_test.copy()

preview[f"{target}\_TRUE"] = y\_test

preview[f"{target}\_PRED"] = y\_pred

preview.head(10)

**b) Predict from a custom input (e.g., from UI or a dict)**

Replace values with realistic inputs for your dataset.

import pandas as pd

import joblib

import json

# Load artifacts

pipe = joblib.load("boston\_linear\_regression.pkl")

with open("linear\_feature\_meta.json", "r") as f:

meta = json.load(f)

all\_feats = meta["all\_features"]

# Example custom feature dict (fill with your own numbers/strings)

example = {

# Classic Boston features example; adjust to your data columns if different

"CRIM": 0.1, "ZN": 18.0, "INDUS": 2.3, "CHAS": 0, "NOX": 0.5,

"RM": 6.2, "AGE": 45.0, "DIS": 4.5, "RAD": 1, "TAX": 300,

"PTRATIO": 17.5, "B": 390.0, "LSTAT": 9.5

}

# Ensure all required columns exist

row = {c: example.get(c, 0.0) for c in all\_feats} # safe default 0.0 / ""

X\_new = pd.DataFrame([row], columns=all\_feats)

pred = pipe.predict(X\_new)[0]

print(f"Predicted {meta['target']}: {pred:.2f} (MEDV is $1000s)")

**B. Streamlit app (Linear Regression only)**

Save as app.py in the same folder as:

* boston\_linear\_regression.pkl
* linear\_feature\_meta.json

import json

import joblib

import pandas as pd

import numpy as np

import streamlit as st

from pathlib import Path

st.set\_page\_config(page\_title="Boston Housing — Linear Regression", page\_icon="🏠", layout="wide")

st.title("🏠 Boston Housing — Linear Regression")

MODEL\_PATH = Path("boston\_linear\_regression.pkl")

META\_PATH = Path("linear\_feature\_meta.json")

@st.cache\_resource

def load\_model():

if not MODEL\_PATH.exists():

st.error("boston\_linear\_regression.pkl not found. Train and save the model first.")

st.stop()

return joblib.load(MODEL\_PATH)

@st.cache\_data

def load\_meta():

if not META\_PATH.exists():

st.error("linear\_feature\_meta.json not found.")

st.stop()

with open(META\_PATH, "r") as f:

return json.load(f)

pipe = load\_model()

meta = load\_meta()

all\_feats = meta["all\_features"]

num\_feats = meta["numeric\_features"]

cat\_feats = meta["categorical\_features"]

target = meta["target"]

# Sidebar inputs

st.sidebar.header("🔢 Enter Features")

inputs = {}

for col in all\_feats:

if col in num\_feats:

inputs[col] = st.sidebar.number\_input(col, value=0.0, step=0.1, format="%.3f")

else:

inputs[col] = st.sidebar.text\_input(col, value="")

X\_user = pd.DataFrame([inputs], columns=all\_feats)

left, right = st.columns([1,1])

with left:

st.subheader("🔮 Single Prediction")

if st.button("Predict"):

try:

pred = pipe.predict(X\_user)[0]

st.success(f"Estimated {target}: \*\*{pred:.2f}\*\* (MEDV is $1000s)")

except Exception as e:

st.error(f"Prediction failed: {e}")

with right:

st.subheader("📦 Batch Predictions (CSV)")

st.caption("CSV must contain exactly these columns (any order):")

st.code(", ".join(all\_feats))

up = st.file\_uploader("Upload CSV", type=["csv"])

if up is not None:

try:

df\_in = pd.read\_csv(up)

missing = [c for c in all\_feats if c not in df\_in.columns]

if missing:

st.error(f"Missing columns: {missing}")

else:

df\_in = df\_in[all\_feats]

preds = pipe.predict(df\_in)

out = df\_in.copy()

out[target + "\_PRED"] = preds

st.dataframe(out.head(20))

st.download\_button(

"Download Predictions CSV",

data=out.to\_csv(index=False).encode("utf-8"),

file\_name="boston\_linear\_predictions.csv",

mime="text/csv"

)

except Exception as e:

st.error(f"Batch prediction failed: {e}")

**requirements.txt**

streamlit

pandas

numpy

scikit-learn

joblib

**Run**

pip install -r requirements.txt

streamlit run app.py