Got it ✅ — you want simple JupyterLab (local) Python scripts that replicate the same machine learning ideas from your SageMaker JumpStart examples, without using SageMaker.

Below are five ready-to-run Python scripts (each standalone, minimal, and using free/open-source libraries). You can paste each into a `.ipynb` cell or run in any Python notebook environment.

🧠 1. Sentiment Analysis of Product Reviews

Goal: Predict if a review is positive or negative.

```python
# Sentiment Analysis with Hugging Face Transformers
!pip install transformers torch --quiet

from transformers import pipeline

# Load pretrained sentiment analysis pipeline
sentiment_model = pipeline("sentiment-analysis")

# Example reviews
reviews = [
    "This product is amazing! I love it.",
    "Terrible quality, broke after one use.",
    "Pretty good for the price.",
    "I wouldn't recommend this to anyone."
]

# Predict sentiment
for review in reviews:
    result = sentiment_model(review)[0]
    print(f"{review} → {result['label']} ({result['score']:.2f})")
```

🧩 Concept learned: NLP, text classification, pre-trained model inference.

🖼️ 2. Image Classification — Cats vs. Dogs

Goal: Classify images using a pretrained model.

```python
# Image classification using a pretrained ResNet
!pip install torchvision torch pillow --quiet

from torchvision import models, transforms
from PIL import Image
```

```python
import torch

# Load pretrained ResNet model
model = models.resnet18(pretrained=True)
model.eval()

# Image preprocessing
preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225]),
])

# Example: classify one local image
img_path = "cat.jpg"  # replace with your image path
img = Image.open(img_path)
input_tensor = preprocess(img).unsqueeze(0)

# Predict
with torch.no_grad():
    outputs = model(input_tensor)
    _, predicted = torch.max(outputs, 1)

# Decode label
from torchvision import datasets
labels = datasets.ImageNet.classes
print("Predicted label:", labels[predicted.item()])
```

🧩 Concept learned: Computer vision, transfer learning, image inference.

📊 3. Predict House Prices (Simple Regression)

Goal: Train a regression model on small tabular data.

```python
# Simple regression with scikit-learn
!pip install scikit-learn pandas --quiet

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Example dataset
data = pd.DataFrame({
```

```
    "size_sqft": [1000, 1500, 2000, 2500, 3000],
    "bedrooms": [2, 3, 3, 4, 4],
    "price": [200000, 250000, 280000, 350000, 400000]
})

X = data[["size_sqft", "bedrooms"]]
y = data["price"]

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Test prediction
print("Predicted price for 1800 sqft, 3 bedrooms:",
      model.predict([[1800, 3]])[0])
```

🧩 Concept learned: Regression, model training, prediction.

📈 4. Sales Forecasting (Time Series)

Goal: Predict future sales using ARIMA.

```
# Simple Time Series Forecasting with statsmodels
!pip install pandas matplotlib statsmodels --quiet

import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA

# Fake sales data
data = pd.DataFrame({
    "date": pd.date_range(start="2024-01-01", periods=20, freq="W"),
    "sales": [100, 120, 130, 150, 170, 160, 180, 200, 210, 230,
              220, 240, 260, 280, 300, 310, 330, 350, 370, 390]
})
data.set_index("date", inplace=True)

# Train ARIMA model
model = ARIMA(data["sales"], order=(1,1,1))
model_fit = model.fit()

# Forecast next 5 weeks
```

```
forecast = model_fit.forecast(steps=5)
print(forecast)

# Plot results
plt.plot(data.index, data["sales"], label="History")
plt.plot(forecast.index, forecast, label="Forecast", linestyle="--")
plt.legend()
plt.show()
```

🧩 Concept learned: Time series forecasting, ARIMA modeling.

🎯 5. Fake News Headline Detection

Goal: Classify headlines as "real" or "fake".

```
# Fake news detection with a simple text classifier
!pip install scikit-learn pandas --quiet

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline

# Example dataset
data = pd.DataFrame({
    "headline": [
        "Government announces new education reforms",
        "Aliens landed in my backyard last night!",
        "Stock market reaches new high this week",
        "Celebrity adopts 12 dragons in secret zoo",
    ],
    "label": ["real", "fake", "real", "fake"]
})

# Split data
X_train, X_test, y_train, y_test = train_test_split(data["headline"], data["label"], test_si

# Build pipeline
model = make_pipeline(TfidfVectorizer(), LogisticRegression())
model.fit(X_train, y_train)

# Test
print("Prediction:", model.predict(["New study shows benefits of green tea"])[0])
print("Accuracy:", model.score(X_test, y_test))
```

Concept learned: Text classification, binary NLP model.