Linear Regression CRISP-DM Project Summary

Simple Linear Regression - CRISP-DM Example This document summarizes the Streamlit-based project for demonstrating linear regression using synthetic data, following CRISP-DM methodology.

core.md Summary

```
# CRISP-DM Summary
1. Business Understanding – define purpose of modeling.
2. Data Understanding – generate synthetic data (y = ax + b + noise).
3. Data Preparation – prepare arrays for model input.
4. Modeling – fit LinearRegression model.
5. Evaluation – compute MSE and R².
6. Deployment – interactive Streamlit app.
```

devLog.md Summary

```
# Development Log
- Created Python environment and installed dependencies.
- Added sliders for slope, intercept, noise, and number of points.
- Built linear regression using sklearn.
- Displayed metrics and visualization in Streamlit.
```

spec.md Summary

```
# Specification
- User-adjustable parameters: slope (a), intercept (b), noise, number of points.
- Model: sklearn LinearRegression.
- Metrics: MSE, R².
- Visualization: scatter + regression line.
- Deployment: Streamlit interactive app.
```

test.md Summary

```
# Test Plan
1. Generate 50 points with slope=2, intercept=5, noise=1 → expect correct ranges.
2. Fit regression → slope ≈ 2, intercept ≈ 5.
3. R² ≥ 0.9 and MSE reasonable.
4. Plot renders correctly.
5. Adjust sliders → app updates dynamically.
```

Streamlit App Code (app.py)

```
import streamlit as st
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

st.title("Simple Linear Regression Explorer - CRISP-DM Example")

st.header("1■■ Business Understanding")
st.write("This app generates a simple linear dataset and performs linear regression.")

st.header("2■■ Data Generation Parameters")
a = st.slider("Slope (a)", min_value=-10.0, max_value=10.0, value=2.0, step=0.1)
b = st.slider("Intercept (b)", min_value=-10.0, max_value=10.0, value=5.0, step=0.1)
```

```
noise = st.slider("Noise level", min_value=0.0, max_value=10.0, value=1.0, step=0.1)
n_points = st.slider("Number of points", min_value=10, max_value=200, value=50, step=1)

st.header("3■■ Data Understanding & Preparation")
x = np.linspace(0, 10, n_points)
y = a*x + b + np.random.normal(0, noise, n_points)
data = pd.DataFrame({"X": x, "Y": y})
st.write(data.head())

st.header("4■■ Modeling")
X = x.reshape(-1,1)
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)

st.write(f"Learned slope: {model.coef_[0]:.3f}")
st.write(f"Learned intercept: {model.intercept_:.3f}")

st.header("5■■ Evaluation")
mse = mean_squared_error(y, y_pred)
r2 = r2_score(y, y_pred)
st.write(f"Mean Squared Error: {mse:.3f}")
st.write(f"R² Score: {r2:.3f}")

st.header("Visualization")
plt.figure(figsize=(8,5))
plt.scatter(x, y, color='blue', label='Data')
plt.plot(x, y_pred, color='red', label='Regression Line')
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Linear Regression Fit")
plt.legend()
st.pyplot(plt)
```