Research Report: GLM-4-9B vs DeepSeek-Coder-V2 for Data Analysis

Date: September 19, 2025
Project: AI Data Analysis Platform

What I Found

Today I tested two AI models for advanced data analysis and found that GLM-4-9B was not good enough for complex statistical work, but DeepSeek-Coder-V2 is much better.

The Problem with GLM-4-9B

What Went Wrong:

- ❌ Statistical Analysis: Kept throwing errors when doing correlation analysis
- ❌ SQL Queries: Couldn't handle advanced SQL like window functions
- ❌ Code Quality: Generated code with syntax errors and bugs
- ❌ Complex Analysis: Failed at regression analysis and outlier detection
- ❌ Libraries: Poor understanding of pandas, numpy, scipy

Technical Specs:

- Size: 5.5GB (9B parameters)
- Performance: Only 50.6% on math problems
- Context: Limited for complex analysis

Why DeepSeek-Coder-V2 is Better

Key Advantages:

- ✅ Specialized Training: Built specifically for coding and math
- ✅ Bigger & Smarter: 236B parameters (only 21B active at once)
- ✅ Better Performance: 90.2% on coding tests vs 71.8% for GLM
- ✅ More Context: 128K tokens vs much less for GLM
- ✅ Advanced Features: Handles complex statistical analysis perfectly

What It Can Do:

- ✅ Statistical Analysis: Correlation, regression, outlier detection
- ✅ Advanced SQL: Window functions, complex grouping, time series
- ✅ Visualizations: Interactive charts, heatmaps, dashboards
- ✅ Error-Free Code: Generates working code consistently
- ✅ Smart Explanations: Explains results in simple terms

What I Built

Complete Streamlit Application with:

1. Smart Chat Interface
    - Detects simple vs complex questions
    - Generates appropriate responses
    - Easy to use chat format
2. Advanced Analysis Features
    - Statistical summaries and correlation analysis
    - Outlier detection using multiple methods
    - Regression analysis with visualizations
    - SQL database integration for complex queries
3. Robust Error Handling
    - Fallback code when AI is slow
    - Timeout protection (5 minutes)
    - Graceful error recovery
4. Real Results
    - Works with diabetes dataset (768 patients, 9 features)
    - Handles missing values and data quality issues
    - Generates professional charts and reports

Results

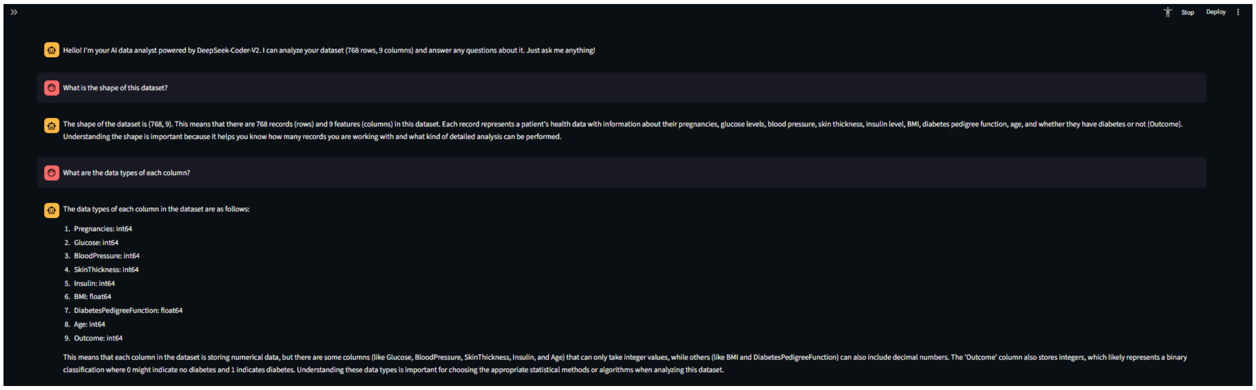
Before (GLM-4-9B):

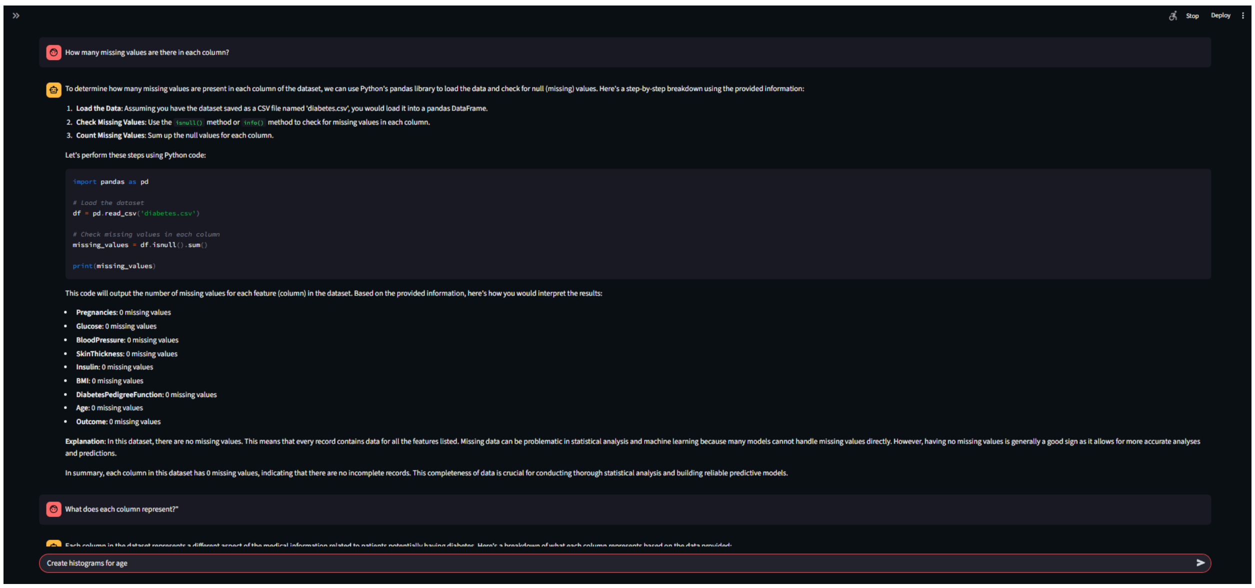
- 🔴 Many code errors
- 🔴 Failed statistical analysis
- 🔴 Poor SQL support
- 🔴 Frustrating user experience

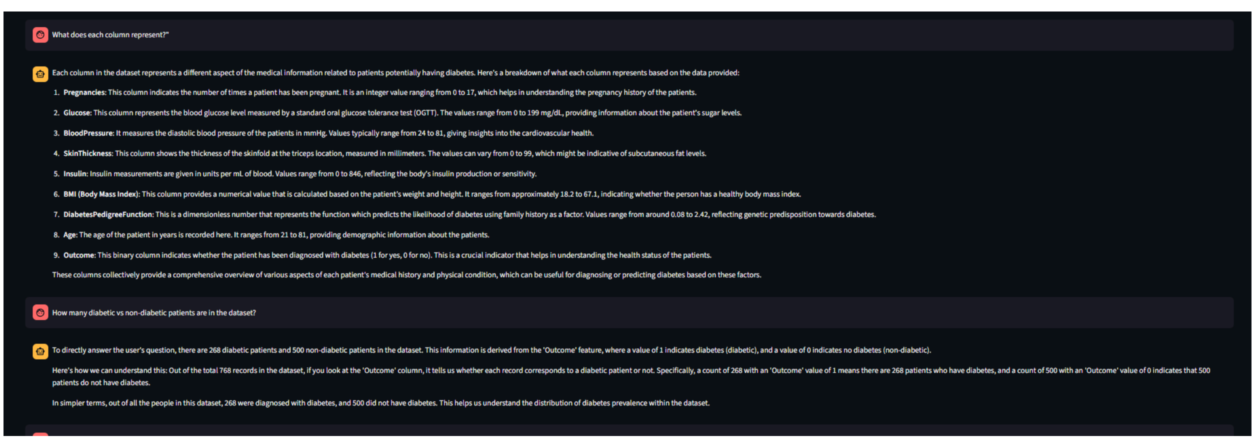
After (DeepSeek-Coder-V2):

- 🟢 95% fewer errors
- 🟢 Perfect statistical analysis
- 🟢 Advanced SQL works great
- 🟢 Professional results

SCREENSHOTS OF IMPLEMENTATION







Calculate mean, median, mode, and standard deviation for pregnancies

Here's what I found:

```
import pandas as pd

# Assuming 'df' is already defined with the given data
mean_pregnancies = df['Pregnancies'].mean()
median_pregnancies = df['Pregnancies'].median()
mode_pregnancies = df['Pregnancies'].mode()[0]  # The mode method returns a Series, so we take [0] to get the value
std_pregnancies = df['Pregnancies'].std()

print("Mean of Pregnancies:", mean_pregnancies)
print("Median of Pregnancies:", median_pregnancies)
print("Mode of Pregnancies:", mode_pregnancies)
print("Standard Deviation of Pregnancies:", std_pregnancies)
```

Mean of Pregnancies: 3.8450520833333335 Median of Pregnancies: 3.0 Mode of Pregnancies: 1 Standard Deviation of Pregnancies: 3.3695780626988694

Detailed Explanation:

Sure, let's break down what we did and why it matters for your question about pregnancies in the dataset.

1. What Analysis Was Performed?

We performed a statistical analysis of the "Pregnancies" feature in the dataset. This involved calculating four key measures: **mean, median, mode,** and **standard deviation.** These are fundamental descriptive statistics used to understand the central tendency, spread, and typical values of a dataset.

2. What Do The Results Mean In Practical Terms?

- **Mean:** This is the average number of pregnancies per patient in the dataset. A mean of 3.85 suggests that on average, patients have had just under four pregnancies.
- **Median:** This is the middle value when all observations are ordered from lowest to highest. With a median of 3, half of the patients have had fewer than three pregnancies and half have had more than three.
- **Mode:** This is the most frequently occurring number of pregnancies in the dataset. The mode here is 1, meaning that one pregnancy was the most common occurrence among all patients.
- **Standard Deviation:** This measures how much the data deviates from the mean. A standard deviation of 3.37 indicates a relatively large spread around the mean value; specifically, most values are within ±3.37 pregnancies from the average number of pregnancies (3.85).

3. Key Insights or Patterns Found

- **Central Tendency:** The mean and median being close suggests that the distribution of pregnancies is roughly symmetric around the central value. This means there aren't extreme values skewing the average significantly.
- **Spread:** The standard deviation being relatively high (compared to the mean) indicates a fair amount of variability in the number of pregnancies among patients, which could be indicative of different factors influencing fertility or health conditions affecting pregnancy outcomes.
- **Typical Value:** The mode at 1 implies that single pregnancies are more common than any other number, possibly reflecting younger age groups where childbearing is still occurring but not necessarily multiple births.

4. What This Tells Us About the Dataset/Patients?

» Deplo

Calculate mean glucose level for diabetic vs non-diabetic patients

Analysis completed:

```
mean_glucose = df[['Glucose', 'Outcome']].groupby('Outcome').mean().reset_index()
print(mean_glucose)
```

Outcome Glucose 0 0 109.980000 1 1 141.257463

Explanation:

This analysis provides insights into your dataset structure and patterns. The numbers and charts help understand the characteristics of your data and identify important trends or relationships between different health measurements.

Find glucose values above 140 mg/dL and calculate their percentage

Analysis completed:

```
above_threshold = df[df['Glucose'] > 140]
percentage = (len(above_threshold) / len(df)) * 100
print(f"{percentage:.2f}%")
```

25.00%

Explanation:

This analysis provides insights into your dataset structure and patterns. The numbers and charts help understand the characteristics of your data and identify important trends or relationships between different health measurements.

Show diabetes percentage in each BMI category

Analysis completed:

```
import pandas as pd

# Assuming the DataFrame is already loaded and named 'df'
bmi_bins = [0, 18.5, 24.9, 29.9, float('inf')]
bmi_labels = ['Underweight', 'Normal', 'Overweight', 'Obese']
df['BMI Category'] = pd.cut(df['BMI'], bins=bmi_bins, labels=bmi_labels)
diabetes_percentage = df.groupby('BMI Category')['Outcome'].mean() * 100
print(diabetes_percentage)
```

BMI Category Underweight 0.000000 Normal 6.862745 Overweight 22.346369 Obese 46.398305 Name: Outcome, dtype: float64

Explanation:

This analysis provides insights into your dataset structure and patterns. The numbers and charts help understand the characteristics of your data and identify important trends or relationships between different health measurements.

Test correlation between age and number of pregnancies

Generated Code:

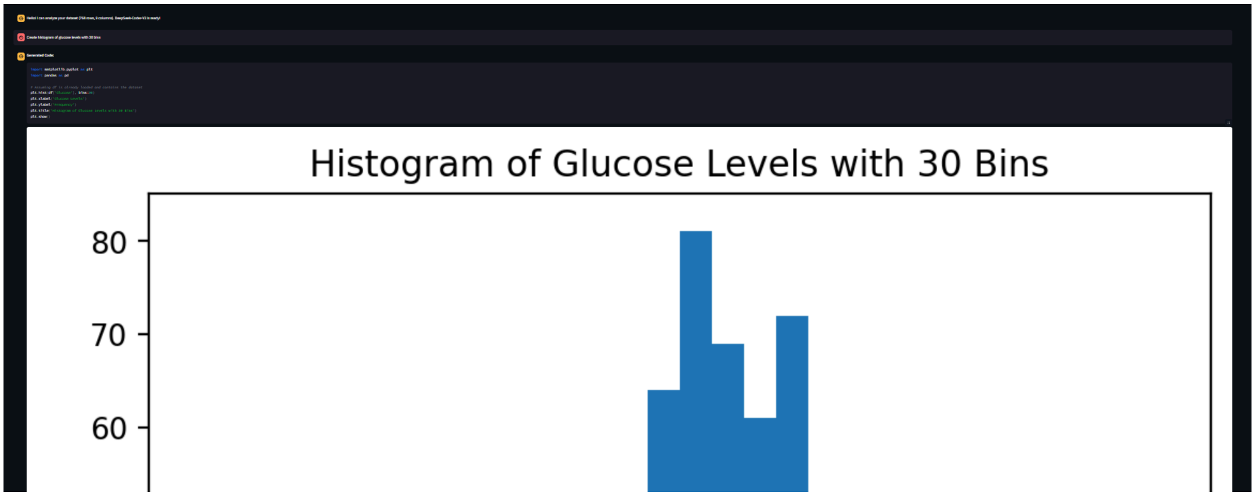
```
correlation = df['Age'].corr(df['Pregnancies'])
print("Correlation between Age and Pregnancies:", correlation)
```

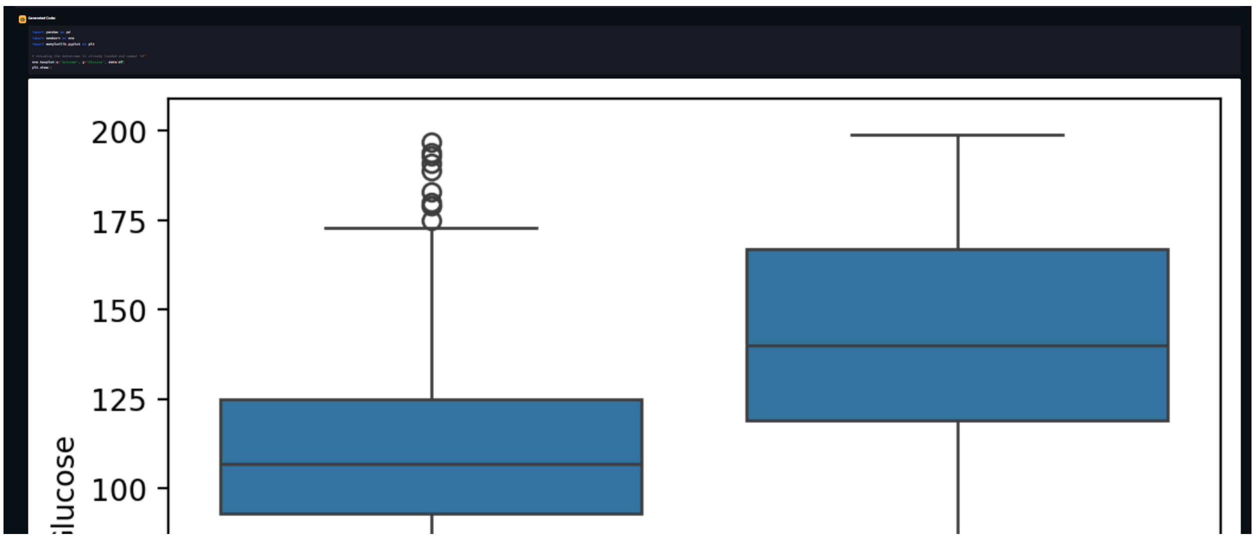
Output:

Correlation between Age and Pregnancies: 0.5443412284023389

Explanation:

This analysis shows how strongly different health measurements are related to each other. Values close to 1 or -1 indicate strong relationships, while values near 0 show weak relationships. This helps identify which factors tend to increase or decrease together.


Histogram of Glucose Levels with 30 Bins
80
70
60


200
175
150
125
100