perplexity

make a code in python to check if i have completed my day by day task for this model

Here's a complete **progress tracker application** to help you monitor your day-by-day learning tasks for the spam filter project:

```
import json
import os
from datetime import datetime

class SpamFilterProgressTracker:
    """
    Track your daily progress on the Spam Filter ML Project
    """

    def __init__(self):
        self.progress_file = "spam_filter_progress.json"
        self.tasks = {
            "Day 1": {
                "title": "Setup and Data Loading",
                "tasks": [
                    "Install required libraries (pandas, scikit-learn, numpy)",
                    "Download spam.csv dataset from Kaggle",
                    "Load dataset using pandas",
                    "Display dataset shape and first few rows",
                    "Check column names and data types"
                ],
                "completed": []
            },
            "Day 2": {
                "title": "Data Cleaning and Preparation",
                "tasks": [
                    "Rename columns to 'label' and 'message'",
                    "Convert labels to binary (spam=1, ham=0)",
                    "Check class distribution (value_counts)",
                    "Understand imbalanced dataset concept",
                    "Display sample messages with labels"
                ],
                "completed": []
            },
            "Day 3": {
                "title": "Text Preprocessing",
                "tasks": [
                    "Create preprocess_text() function",
                    "Implement lowercase conversion",
                    "Remove URLs from text",
                    "Remove punctuation",
                    "Remove numbers",
                    "Remove extra whitespace",
                    "Apply preprocessing to all messages",
                    "Compare original vs cleaned messages"
                ],
                "completed": []
            },
            "Day 4": {
                "title": "Feature Engineering - TF-IDF",
                "tasks": [
                    "Understand TF-IDF concept",
                    "Create TfidfVectorizer with max_features=3000",
                    "Fit and transform cleaned messages",
                    "Create feature matrix X",
                    "Extract labels as y",
                    "Check feature matrix shape",
                    "View sample feature names"
                ],
                "completed": []
            },
            "Day 5": {
                "title": "Train-Test Split and Model Training",
                "tasks": [
                    "Import train_test_split",
                    "Split data (80% train, 20% test)",
                    "Use stratify parameter",
                    "Verify train and test set sizes",
                    "Import MultinomialNB",
                    "Create Naive Bayes model",
                    "Train model using fit()",
                    "Understand Naive Bayes probability concept"
                ],
                "completed": []
            },
            "Day 6": {
                "title": "Model Evaluation",
                "tasks": [
                    "Make predictions on test set",
                    "Calculate accuracy score",
                    "Create confusion matrix",
                    "Understand TP, TN, FP, FN",
                    "Generate classification report",
                    "Understand Precision metric",
                    "Understand Recall metric",
                    "Understand F1-Score",
                    "Analyze model strengths and weaknesses"
                ],
                "completed": []
            },
            "Day 7": {
                "title": "Testing and Saving Model",
                "tasks": [
                    "Create predict_spam() function",
                    "Test with custom spam messages",
                    "Test with custom ham messages",
                    "Get prediction probabilities",
                    "Save model using pickle",
                    "Save vectorizer using pickle",
                    "Create load_and_predict() function",
                    "Test loading and predicting",
                    "Document your project"
                ],
                "completed": []
            }
        }
        self.load_progress()

    def load_progress(self):
        """Load saved progress from JSON file"""
        if os.path.exists(self.progress_file):
            with open(self.progress_file, 'r') as f:
                saved_data = json.load(f)
                # Merge saved progress with current tasks
                for day, data in saved_data.items():
                    if day in self.tasks:
                        self.tasks[day]['completed'] = data.get('completed', [])

    def save_progress(self):
        """Save current progress to JSON file"""
        with open(self.progress_file, 'w') as f:
            json.dump(self.tasks, f, indent=4)
        print("✓ Progress saved!")

    def display_all_tasks(self):
        """Display all tasks for the week"""
        print("\n" + "="*70)
        print("□ SPAM FILTER ML PROJECT - 7 DAY LEARNING PLAN")
        print("="*70)

        for day, data in self.tasks.items():
            total_tasks = len(data['tasks'])
            completed_tasks = len(data['completed'])
            percentage = (completed_tasks / total_tasks * 100) if total_tasks > 0 else 0

            # Progress bar
            bar_length = 30
            filled = int(bar_length * completed_tasks / total_tasks)
            bar = "█" * filled + "░" * (bar_length - filled)

            print(f"\n{day}: {data['title']}")
            print(f"Progress: [{bar}] {percentage:.0f}% ({completed_tasks}/{total_tasks})
            print("-" * 70)

            for i, task in enumerate(data['tasks'], 1):
                status = "✓" if i in data['completed'] else "☐"
                print(f"  {status} {i}. {task}")

        print("\n" + "="*70)

    def display_day_tasks(self, day):
        """Display tasks for a specific day"""
        if day not in self.tasks:
            print(f"✖ Invalid day! Please use Day 1 to Day 7")
            return

        data = self.tasks[day]
        total_tasks = len(data['tasks'])
        completed_tasks = len(data['completed'])

        print(f"\n{'='*70}")
        print(f"{day}: {data['title']}")
        print(f"{'='*70}")
        print(f"Progress: {completed_tasks}/{total_tasks} tasks completed\n")

        for i, task in enumerate(data['tasks'], 1):
            status = "✅ DONE" if i in data['completed'] else "□ TODO"
            print(f"{i}. [{status}] {task}")

        print(f"{'='*70}\n")

    def mark_task_complete(self, day, task_number):
        """Mark a specific task as complete"""
        if day not in self.tasks:
            print(f"✖ Invalid day! Please use Day 1 to Day 7")
            return

        data = self.tasks[day]
        if task_number < 1 or task_number > len(data['tasks']):
            print(f"✖ Invalid task number! {day} has {len(data['tasks'])} tasks")
            return

        if task_number not in data['completed']:
            data['completed'].append(task_number)
            data['completed'].sort()
            self.save_progress()
            print(f"✅ Task {task_number} marked as complete!")

            # Check if day is complete
            if len(data['completed']) == len(data['tasks']):
                print(f"□ Congratulations! You've completed {day}!")
        else:
            print(f"⚠  Task {task_number} is already marked as complete")

    def mark_task_incomplete(self, day, task_number):
        """Mark a specific task as incomplete"""
        if day not in self.tasks:
            print(f"✖ Invalid day! Please use Day 1 to Day 7")
            return

        data = self.tasks[day]
        if task_number in data['completed']:
            data['completed'].remove(task_number)
            self.save_progress()
            print(f"□ Task {task_number} marked as incomplete")
        else:
            print(f"⚠  Task {task_number} is already incomplete")

    def mark_day_complete(self, day):
        """Mark all tasks in a day as complete"""
        if day not in self.tasks:
            print(f"✖ Invalid day! Please use Day 1 to Day 7")
            return

        data = self.tasks[day]
        data['completed'] = list(range(1, len(data['tasks']) + 1))
        self.save_progress()
        print(f"□ All tasks for {day} marked as complete!")

    def get_overall_progress(self):
        """Calculate overall project completion"""
        total_tasks = sum(len(data['tasks']) for data in self.tasks.values())
        completed_tasks = sum(len(data['completed']) for data in self.tasks.values())
        percentage = (completed_tasks / total_tasks * 100) if total_tasks > 0 else 0

        print(f"\n{'='*70}")
        print(f"□ OVERALL PROJECT PROGRESS")
        print(f"{'='*70}")
        print(f"Total Tasks: {total_tasks}")
        print(f"Completed: {completed_tasks}")
        print(f"Remaining: {total_tasks - completed_tasks}")
        print(f"Progress: {percentage:.1f}%")

        # Progress bar
        bar_length = 50
        filled = int(bar_length * completed_tasks / total_tasks)
        bar = "█" * filled + "░" * (bar_length - filled)
        print(f"\n[{bar}] {percentage:.1f}%")

        if percentage == 100:
            print("\n□ CONGRATULATIONS! You've completed the entire project! □")
        elif percentage >= 75:
            print("\n□ Great progress! You're almost there!")
        elif percentage >= 50:
            print("\n□ Keep going! You're halfway through!")
        elif percentage >= 25:
            print("\n□ Good start! Keep up the momentum!")
        else:
            print("\n□ Just getting started! You got this!")

        print(f"{'='*70}\n")

    def get_next_task(self):
        """Show the next incomplete task"""
        for day, data in self.tasks.items():
            for i, task in enumerate(data['tasks'], 1):
                if i not in data['completed']:
                    print(f"\n{'='*70}")
                    print(f"□ NEXT TASK TO COMPLETE")
                    print(f"{'='*70}")
                    print(f"Day: {day}")
                    print(f"Task {i}: {task}")
                    print(f"{'='*70}\n")
                    return
        print("□ All tasks completed! Great job!")

    def reset_progress(self):
        """Reset all progress"""
        confirm = input("⚠  Are you sure you want to reset ALL progress? (yes/no): ")
        if confirm.lower() == 'yes':
            for data in self.tasks.values():
                data['completed'] = []
            self.save_progress()
            print("✓ All progress has been reset")
        else:
            print("✖ Reset cancelled")


def main():
    """Main menu for the progress tracker"""
    tracker = SpamFilterProgressTracker()

    while True:
        print("\n" + "="*70)
        print("□ SPAM FILTER ML PROJECT TRACKER")
        print("="*70)
        print("1.  View all tasks")
        print("2.  View specific day tasks")
        print("3.  Mark task as complete")
        print("4.  Mark task as incomplete")
        print("5.  Mark entire day as complete")
        print("6.  View overall progress")
        print("7.  Show next task")
        print("8.  Reset all progress")
        print("9.  Exit")
        print("="*70)

        choice = input("\nEnter your choice (1-9): ").strip()

        if choice == '1':
            tracker.display_all_tasks()

        elif choice == '2':
            day = input("Enter day (e.g., Day 1): ").strip()
            tracker.display_day_tasks(day)

        elif choice == '3':
            day = input("Enter day (e.g., Day 1): ").strip()
            try:
                task_num = int(input("Enter task number: ").strip())
                tracker.mark_task_complete(day, task_num)
            except ValueError:
                print("✖ Please enter a valid number")

        elif choice == '4':
            day = input("Enter day (e.g., Day 1): ").strip()
            try:
                task_num = int(input("Enter task number: ").strip())
                tracker.mark_task_incomplete(day, task_num)
            except ValueError:
                print("✖ Please enter a valid number")

        elif choice == '5':
            day = input("Enter day (e.g., Day 1): ").strip()
            tracker.mark_day_complete(day)

        elif choice == '6':
            tracker.get_overall_progress()

        elif choice == '7':
            tracker.get_next_task()

        elif choice == '8':
            tracker.reset_progress()

        elif choice == '9':
            print("\n Good luck with your learning! Keep going!")
            break

        else:
            print("✖ Invalid choice. Please select 1-9")


if __name__ == "__main__":
    main()
```

How to Use This Tracker:

1. Save and Run:

```
# Save as: spam_filter_tracker.py
python spam_filter_tracker.py
```

2. Menu Options:

Option 1: View All Tasks

```
 SPAM FILTER ML PROJECT - 7 DAY LEARNING PLAN
==================================================================

Day 1: Setup and Data Loading
Progress: [████████████░░░░░░░░░░░░░░░░░░] 40% (2/5)
------------------------------------------------------------------
  ✓ 1. Install required libraries (pandas, scikit-learn, numpy)
  ✓ 2. Download spam.csv dataset from Kaggle
  ☐ 3. Load dataset using pandas
  ☐ 4. Display dataset shape and first few rows
  ☐ 5. Check column names and data types
```

Option 2: View Specific Day

```
Enter day: Day 3

======================================================================
Day 3: Text Preprocessing
======================================================================
Progress: 5/8 tasks completed

1. [✅ DONE] Create preprocess_text() function
2. [✅ DONE] Implement lowercase conversion
3. [✅ DONE] Remove URLs from text
4. [✅ DONE] Remove punctuation
5. [✅ DONE] Remove numbers
6. [▯ TODO] Remove extra whitespace
7. [▯ TODO] Apply preprocessing to all messages
8. [▯ TODO] Compare original vs cleaned messages
======================================================================
```

Option 3: Mark Task Complete

```
Enter day: Day 1
Enter task number: 3
✅ Task 3 marked as complete!
```

Option 6: View Overall Progress

```
======================================================================
▯ OVERALL PROJECT PROGRESS
======================================================================
Total Tasks: 55
Completed: 28
Remaining: 27
Progress: 50.9%

[█████████████████████████░░░░░░░░░░░░░░░░░░░░░░░░░] 50.9%

▯ Keep going! You're halfway through!
======================================================================
```

Option 7: Show Next Task

```
======================================================================
▯ NEXT TASK TO COMPLETE
======================================================================
Day: Day 3
Task 6: Remove extra whitespace
======================================================================
```

Features:

✅ **Persistent Progress** - Saves to `spam_filter_progress.json` file
✅ **Visual Progress Bars** - See completion percentage at a glance
✅ **Day-by-Day Breakdown** - Track each day separately
✅ **Overall Statistics** - Monitor total project progress
✅ **Next Task Suggestion** - Never lose your place
✅ **Motivational Messages** - Get encouragement as you progress
✅ **Easy Task Management** - Mark complete/incomplete with numbers
✅ **Reset Option** - Start fresh if needed

Quick Commands:

```
# Just want to check today's tasks?
tracker = SpamFilterProgressTracker()
tracker.display_day_tasks("Day 1")

# Mark a task done after completing it
tracker.mark_task_complete("Day 1", 3)

# See what to do next
tracker.get_next_task()

# Check overall progress
tracker.get_overall_progress()
```

Progress File Example:

The tracker saves your progress in `spam_filter_progress.json`:

```
{
    "Day 1": {
        "title": "Setup and Data Loading",
        "tasks": [...],
        "completed": [1, 2, 3]
    },
    "Day 2": {
        "title": "Data Cleaning and Preparation",
        "tasks": [...],
        "completed": [1, 2, 3, 4, 5]
    }
}
```

This tracker will help you stay organized and motivated throughout your one-week spam filter learning journey! 🚀