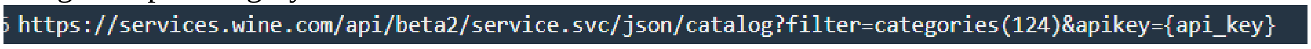
Weekly progress report – March 9 (5th report)

Still working on: implementing a wine recommendation engine(training ML models is in process)

The recommendation engine using machine learning algorithms to analyze data such as purchase history, reviews, ratings, and even social media activity, to identify patterns in customer preferences and suggest wines that are likely to appeal to them.

1. Data Collection: The first step is to collect data on wines. The data should include information such as wine type, region, grape variety, price, and ratings.

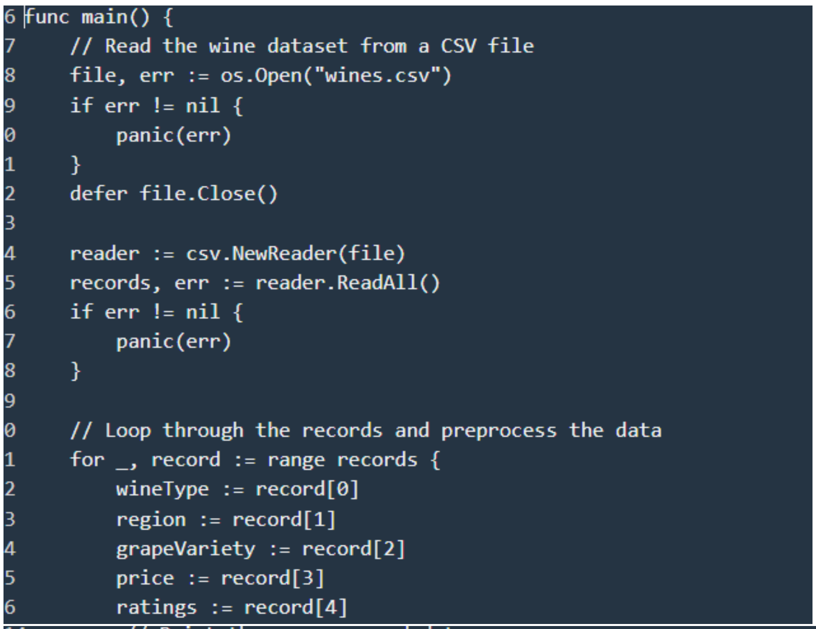
How I'm collecting the data? I'm integrating wine.com API.
The API endpoints aren't the most intuitive to use. If you're accustomed to URLs such as api.example.com/wines (list all wines) and api.example.com/api/wines/1234 (information about wine with the ID of 1234) you won't find them here. So here's the example on how to start querying the Catalog API for wines using multiple category combinations:



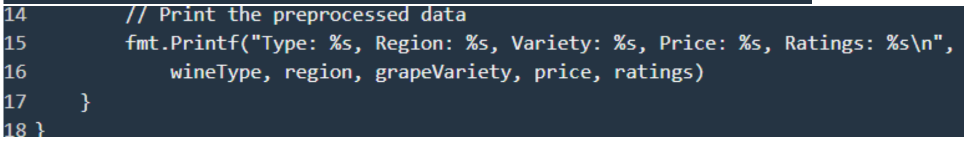
```
https://services.wine.com/api/beta2/service.svc/json/catalog?filter=categories(124)&apikey={api_key}
```

2. Data Preprocessing: Once I have collected the data, i'm preprocessing it. This involves cleaning and transforming the data into a format that can be easily used by the recommendation engine.

Now let's say we have a dataset of wines in CSV format with columns for wine type, region, grape variety, price, and ratings. We will use the Golang "encoding/csv" package to read the data and preprocess it.



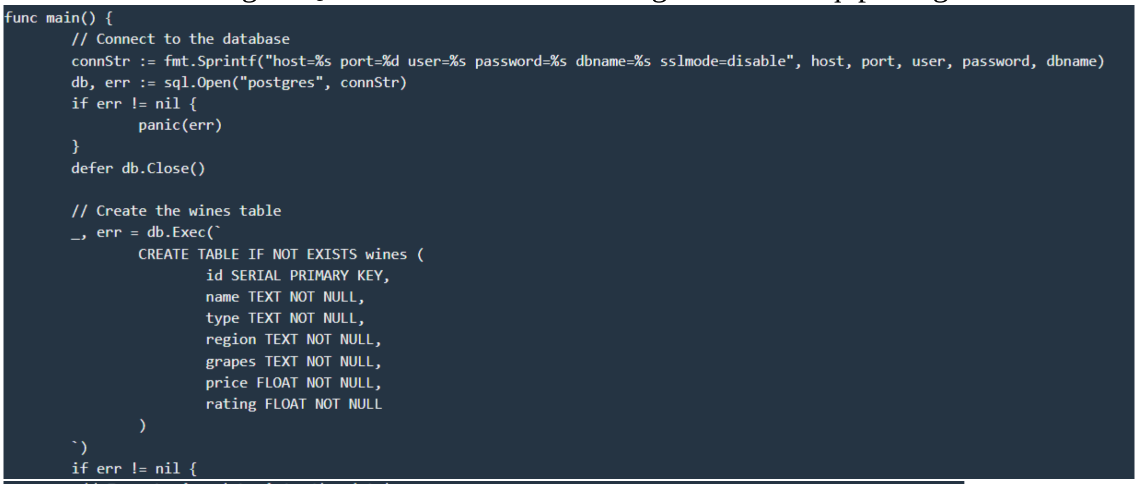
```
func main() {
    // Read the wine dataset from a CSV file
    file, err := os.Open("wines.csv")
    if err != nil {
        panic(err)
    }
    defer file.Close()

    reader := csv.NewReader(file)
    records, err := reader.ReadAll()
    if err != nil {
        panic(err)
    }

    // Loop through the records and preprocess the data
    for _, record := range records {
        wineType := record[0]
        region := record[1]
        grapeVariety := record[2]
        price := record[3]
        ratings := record[4]
```



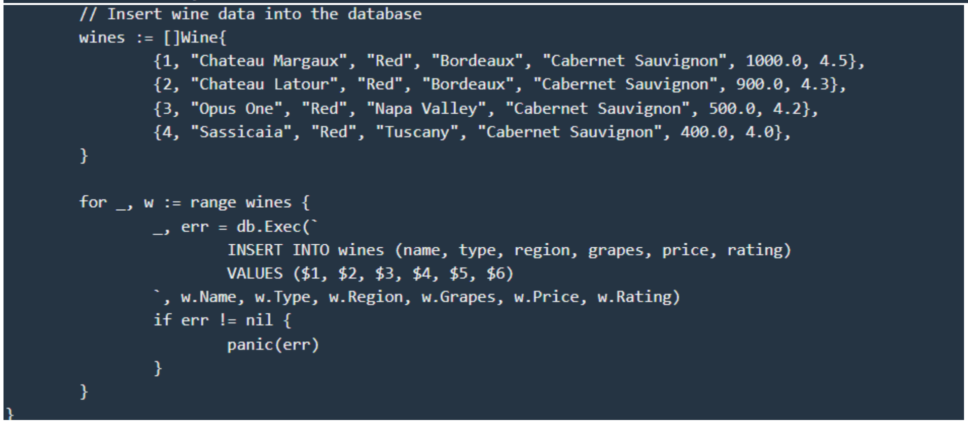
```
        // Print the preprocessed data
        fmt.Printf("Type: %s, Region: %s, Variety: %s, Price: %s, Ratings: %s\n",
            wineType, region, grapeVariety, price, ratings)
    }
}
```

3. Data Storage: After preprocessing the data, I need to store it in a database.

I will create a PostgreSQL database and use Golang's database/sql package to interact with it.



```go
func main() {
	// Connect to the database
	connStr := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s sslmode=disable", host, port, user, password, dbname)
	db, err := sql.Open("postgres", connStr)
	if err != nil {
		panic(err)
	}
	defer db.Close()

	// Create the wines table
	_, err = db.Exec(`
		CREATE TABLE IF NOT EXISTS wines (
			id SERIAL PRIMARY KEY,
			name TEXT NOT NULL,
			type TEXT NOT NULL,
			region TEXT NOT NULL,
			grapes TEXT NOT NULL,
			price FLOAT NOT NULL,
			rating FLOAT NOT NULL
		)
	`)
	if err != nil {
```



```go
	// Insert wine data into the database
	wines := []Wine{
		{1, "Chateau Margaux", "Red", "Bordeaux", "Cabernet Sauvignon", 1000.0, 4.5},
		{2, "Chateau Latour", "Red", "Bordeaux", "Cabernet Sauvignon", 900.0, 4.3},
		{3, "Opus One", "Red", "Napa Valley", "Cabernet Sauvignon", 500.0, 4.2},
		{4, "Sassicaia", "Red", "Tuscany", "Cabernet Sauvignon", 400.0, 4.0},
	}

	for _, w := range wines {
		_, err = db.Exec(`
			INSERT INTO wines (name, type, region, grapes, price, rating)
			VALUES ($1, $2, $3, $4, $5, $6)
		`, w.Name, w.Type, w.Region, w.Grapes, w.Price, w.Rating)
		if err != nil {
			panic(err)
		}
	}
}
```

I stopped on 4th step while calculating users similarity. I think i'll continue it later and come back with trained ML models next week. See you in 6th report.

4. Recommendation Algorithm: need to choose a recommendation algorithm. There are several recommendation algorithms such as collaborative filtering, content-based filtering, and hybrid filtering. Guess i'm gonna use collaborative filtering.
5. Development of Recommendation Engine: Once i have chosen an algorithm, i need to develop the recommendation engine. I need to choose one of them to work with: gorilla/mux, gin-gonic/gin, and revel/revel. Or maybe I'll just get lazy and remove this part.