Model Development Phase Template

Date	18 June 2025
Team ID	SWTID1749631993
Project Title	Restaurant Recommendation System
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report:

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

```
df_percent = zomato_df[zomato_df['rate'] > 3.5].copy()

# Clean text: fill NaN with empty string and ensure all entries are strings
df_percent.loc[:, 'reviews_list'] = df_percent['reviews_list'].fillna('').astype(str)

# Set index and prepare for similarity calculation
df_percent.set_index('name', inplace=True)
indices = pd.Series(df_percent.index)

# TF-IDF Vectorization
tfidf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=1, stop_words='english')
tfidf_matrix = tfidf.fit_transform(df_percent['reviews_list'])

# Cosine similarity
cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
```

```python
from sklearn.neighbors import NearestNeighbors
def recommend(restaurant_name, top_n=10, df=zomato_df):
    """
    Recommend similar restaurants based on cuisines using TF-IDF and cosine similarity.
    """
    try:
        # 1. Filter dataset (optional memory optimization)
        df_subset = df[df['rate'] > 1.5].copy() if len(df) > 10000 else df.copy()

        # 2. Vectorize cuisines using TF-IDF
        vectorizer = TfidfVectorizer(stop_words='english')
        tfidf_matrix = vectorizer.fit_transform(df_subset['cuisines'].fillna(''))

        # 3. Fit Nearest Neighbors
        nn = NearestNeighbors(n_neighbors=top_n+1, algorithm='brute', metric='cosine')
        nn.fit(tfidf_matrix)

        # 4. Index lookup
        indices = pd.Series(df_subset.index, index=df_subset['name']).drop_duplicates()
        idx = indices.get(restaurant_name)

        if idx is None:
            print(f"❌ Restaurant '{restaurant_name}' not found.")
            suggestions = df[df['name'].str.contains(restaurant_name.split()[0], case=False)]['name'].unique()[:5]
            print("Did you mean one of these?\n", suggestions)
            return None

        # 5. Get similar restaurants
        distances, neighbor_indices = nn.kneighbors(tfidf_matrix[idx])

        # 6. Format and return top N (excluding self)
        recommendations = df_subset.iloc[neighbor_indices[0][1:top_n+1]][
            ['name', 'cuisines', 'rate', 'cost']
        ].sort_values('rate', ascending=False)
```

```python
        recommendations = recommendations.rename(columns={
            'name': 'Restaurant',
            'cuisines': 'Cuisines',
            'rate': 'Mean Rating',
            'cost': 'Cost'
        })

        print(f"\n📍 TOP {top_n} RESTAURANTS LIKE {restaurant_name} WITH SIMILAR REVIEWS:\n")

        return recommendations

    except Exception as e:
        print(f"⚠️ Error: {str(e)}")
        print("Try reducing the dataset size or using a machine with more memory.")
        return None
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
TF-IDF vectorization	-	-	-

Recommendation System using TF-IDF and Nearest Neighbors	-	-	-

Since our system uses unsupervised recommendation based on TF-IDF and cosine similarity, traditional evaluation metrics like classification accuracy or confusion matrix do not apply