EXPRESSEATS

- Food Recipe Recommendation System

PROJECT REPORT

Course: DATA MINING (CMPE-255)

Course Instructor: VIJAY ERANTI

Project Team:

Harshith Akkapelli

Neeharika Singh

Sanjay Bhargav Kudupudi

Github : https://github.com/ExpressNesters/ExpressEats.git

Colab :

Part1: https://colab.research.google.com/drive/1xDXfK0uYqlIlknKVtaToKmxSuUHSyhGb?usp=sharing

Part-2: https://colab.research.google.com/drive/1i6yGBU2sgiFkLy11uICRgaliG6m-6Jh6?usp=sharing

Part-3: https://colab.research.google.com/drive/1WENxYP60Bb-1gnp_w8rDnKdTvaHQu-C0?usp=sharing

Part-4: https://colab.research.google.com/drive/1WENxYP60Bb-1gnp_w8rDnKdTvaHQu-C0?usp=sharing

Dataset: https://www.kaggle.com/datasets/kanishk307/6000-indian-food-recipes-dataset

Abstract:

Have you ever been in a dilemma about what to cook with the ingredients you have with limited time? Addressing a common pain point in our daily lives: not knowing what to prepare with limited time and odd supplies in the kitchen. This can be exhausting because we could be spending our time on more important things. We developed a system called "ExpressEats," which recommends personalized recipes depending on the user's time restrictions and available ingredients.

This project's goal is to create a Food meal Recommendation System that gives users individualized meal recommendations based on common ingredients and short preparation times. This system will combine data science and machine learning approaches to produce a user-friendly application that enhances the cooking experience and assists users in effectively preparing excellent meals.

Introduction:

Our objective is to develop a system that, given the total time and translated ingredients, recommends a recipe that is closest to these criteria. We want to achieve this using the power of data mining and machine learning to build a recipe recommendation system, which take ingredients as the input from the user and utilizes machine learning and recommends the recipes available based on the ingredients provided by the user. By doing so we were able to achieve our objectives.

Key Features:

Users can give ingredients as input.
Users can get the recipe based on the ingredients provided.

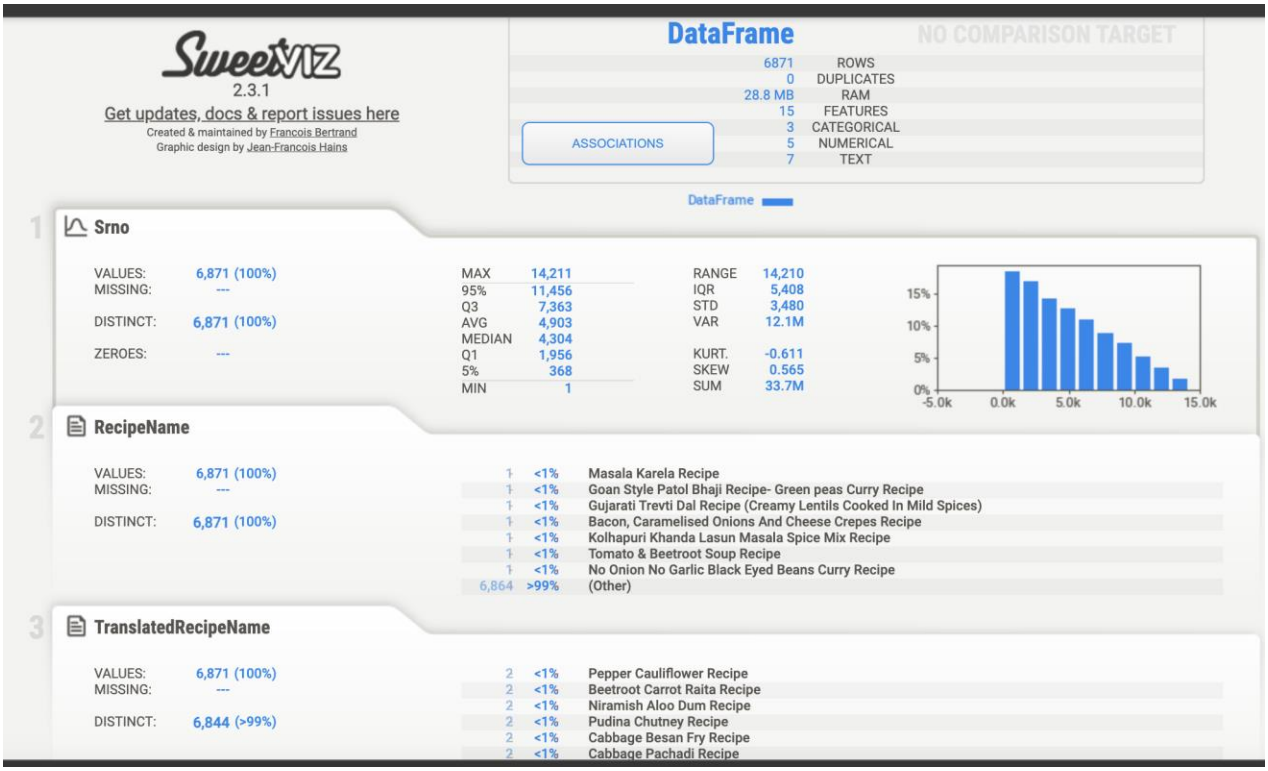
Related Works:

We came across some related recommendation systems, some of them used different approaches and some of them even used complex approaches . Although we have many different techniques, we choose optimal and less complex technical approach. Which differs our work from others.

Data:

We have used a dataset from kaagle which contains data that is required for us to perform and use for our application.

Details of the dataset are below



MISSING:	6 (<1%)	2	<1%
		2	<1%
DISTINCT:	6,856 (>99%)	2	<1%
		2	<1%
		2	<1%
		2	<1%
		2	<1%
		6,851	>99%

1 cup Broken Wheat (Dalia/ Godumai Rava),Salt - to taste,1/2 teaspoon Sunflower Oil,1/3 cup Gram flou...
300 grams Kaddu (Parangikai/ Pumpkin),1 Tamarind - half the size of lemon,1/2 cup Water,1 teaspoon ...
16 Whole Wheat Brown Bread,3 Tomatoes - chopped small,2 Onions - thinly sliced,2 Green Chillies - fin...
500 grams Button mushrooms - chopped,2 Green Chillies - chopped,2 Dry Red Chillies,1 teaspoon Whol...
12 Button mushrooms - (button or crimini variety),2 teaspoon Extra Virgin Olive Oil - for cooking,1/2 cu...
4 आलू,1/2 छोटा चमच्च हींग,1 छोटा चमच्च जीरा,3 सुखी लाल मिर्च,3 बड़े चमच्च अचार,2 बड़े चमच्च सिरका,1/2 छोटा चमच्च श...
(Other)

5 TranslatedIngredients

VALUES:	6,865 (>99%)	2	<1%
MISSING:	6 (<1%)	2	<1%
		2	<1%
DISTINCT:	6,857 (>99%)	2	<1%
		2	<1%
		2	<1%
		2	<1%
		2	<1%
		6,851	>99%

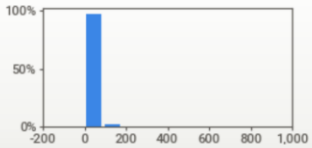
300 grams Kaddu (Parangikai/ Pumpkin),1 Tamarind - half the size of lemon,1/2 cup Water,1 teaspoon ...
1 cup Broken Wheat (Dalia/ Godumai Rava),Salt - to taste,1/2 teaspoon Sunflower Oil,1/3 cup Gram flou...
16 Whole Wheat Brown Bread,3 Tomatoes - chopped small,2 Onions - thinly sliced,2 Green Chillies - fin...
4 sprig Tulsi (holy basil),2 Betel leaves (paan),6 cloves Garlic,1/2 teaspoon Ajwain (Carom seeds),1/3 t...
500 grams Button mushrooms - chopped,2 Green Chillies - chopped,2 Dry Red Chillies,1 teaspoon Whol...
4 आलू,1/2 छोटा चमच्च हींग,1 छोटा चमच्च जीरा,3 सुखी लाल मिर्च,3 बड़े चमच्च अचार,2 बड़े चमच्च सिरका,1/2 छोटा चमच्च श...
2 आलू - काट ले,1 कप गोभी - काट ले,1 कप मेथी - धो कर काट ले,तेल - प्रयोग अनुसार,1 छोटा चमच्च जीरा,हींग - चुटकी भर,1 ...
(Other)

6 PrepTimeInMins

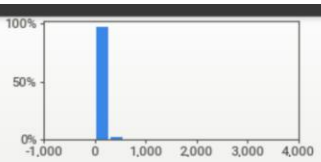
VALUES:	6,871 (100%)	MAX	2,880	RANGE	2,880
MISSING:	---	95%	68	IQR	10.0
		Q3	20	STD	81.0
DISTINCT:	86 (1%)	AVG	29	VAR	6,568
		MEDIAN	15		
ZEROES:	127 (2%)	Q1	10	KURT.	284
		5%	5	SKEW	12.3
		MIN	0	SUM	196k

7 CookTimeInMins

VALUES:	6,871 (100%)	MAX	900	RANGE	900
MISSING:	---	95%	60	IQR	15.0
		Q3	35	STD	34.0
DISTINCT:	67 (<1%)	AVG	31	VAR	1,157
		MEDIAN	30		
ZEROES:	235 (3%)	Q1	20	KURT.	170
		5%	5	SKEW	10.4
		MIN	0	SUM	212k

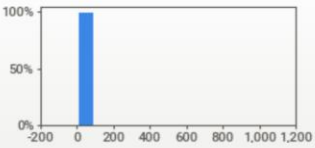


VALUES:	6,871 (100%)	MAX	2,925	RANGE	2,925
MISSING:	---	95%	150	IQR	25.0
		Q3	55	STD	88.7
DISTINCT:	167 (2%)	AVG	59	VAR	7,868
		MEDIAN	40		
ZEROES:	3 (<1%)	Q1	30	KURT.	205
		5%	15	SKEW	10.1
		MIN	0	SUM	408k



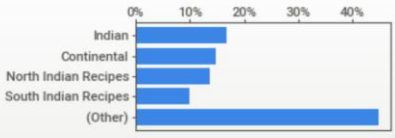
9 Servings

VALUES:	6,871 (100%)	MAX	1,000	RANGE	999
MISSING:	---	95%	8	IQR	0.00
		Q3	4	STD	26.2
DISTINCT:	33 (<1%)	AVG	6	VAR	688
		MEDIAN	4		
ZEROES:	---	Q1	4	KURT.	602
		5%	2	SKEW	22.5
		MIN	1	SUM	38,556



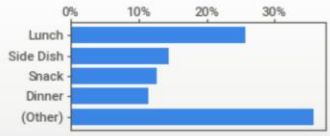
10 Cuisine

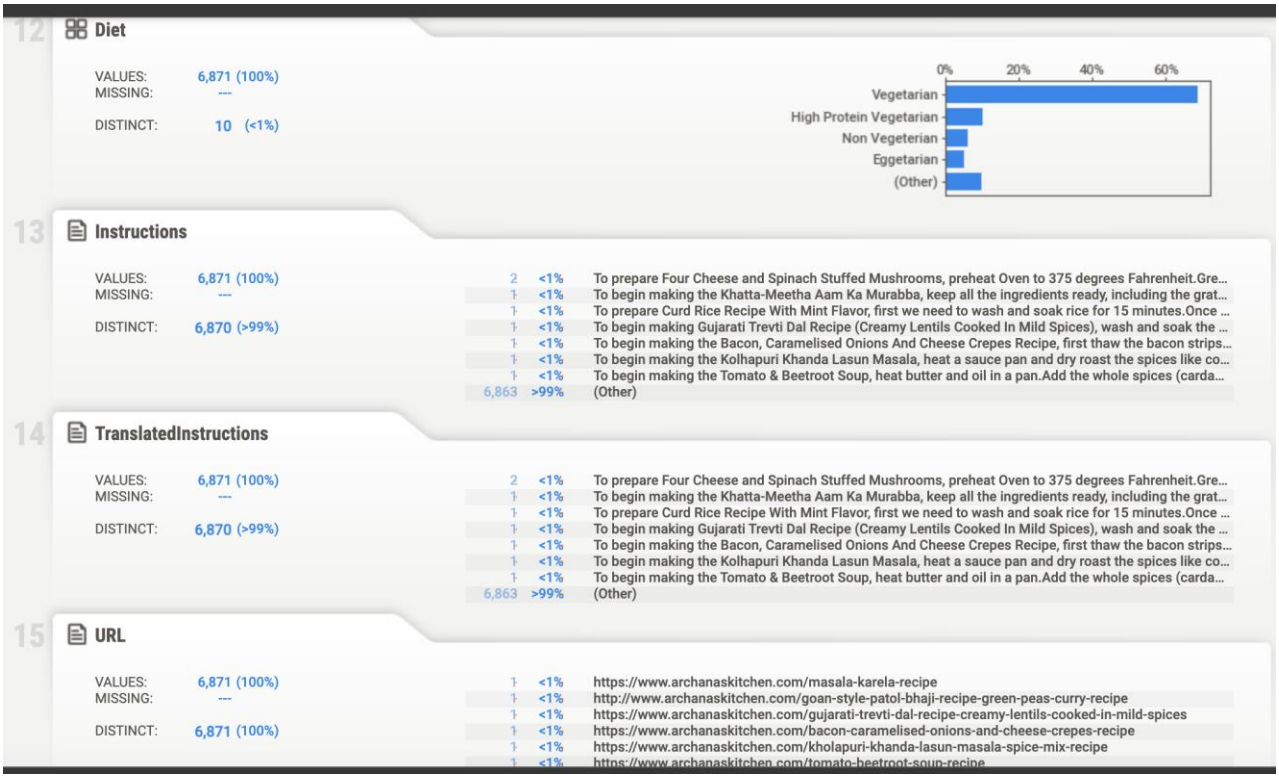
VALUES:	6,871 (100%)
MISSING:	---
DISTINCT:	82 (1%)



11 Course

VALUES:	6,871 (100%)
MISSING:	---
DISTINCT:	20 (<1%)





Methodology:

We have used CRISP-DM methodology for the recommendation system as it has perfectly fit the requirements for modeling our solution.

Business Understanding

This can be viewed as a recommendation system problem where the input features are time and ingredients, and the output is a recipe recommendation.

1. **Data Collection:**

   We have collected a dataset which includes ingredients and cooking time. This dataset has more than 6000+ recipes in it.

2. Data Preprocessing:

This dataset being a vast dataset with more data in it. We have performed data cleaning and preprocessing the data by handling the missing values.

Handle Missing Values

```
[ ] # Dropping rows where 'TranslatedIngredients' or 'TotalTimeInMins' are missing
    data_cleaned = data.dropna(subset=['TranslatedIngredients', 'TotalTimeInMins'])
```

Feature Engineering

```
[ ] # Counting the number of ingredients in each recipe
    data_cleaned['IngredientsCount'] = data_cleaned['TranslatedIngredients'].apply(lambda x: len(x.split(',')))
```

```
/tmp/ipykernel_19/2568196131.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_cleaned['IngredientsCount'] = data_cleaned['TranslatedIngredients'].apply(lambda x: len(x.split(',')))
```

Feature Selection

```
[ ] # Selecting relevant features
    features = data_cleaned[['TotalTimeInMins', 'TranslatedIngredients', 'IngredientsCount']]
```

Splitting the Data

```
from sklearn.neighbors import NearestNeighbors
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
data.dropna(subset=['TotalTimeInMins', 'TranslatedIngredients', 'TranslatedRecipeName'], inplace=True)

# Vectorize the ingredients
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(data['TranslatedIngredients'])
joblib.dump(tfidf, 'tfidf_vectorizer.pkl')
# Normalize TotalTimeInMins
max_time = data['TotalTimeInMins'].max()
normalized_time = data['TotalTimeInMins'] / max_time

# Combine time with ingredient vectors
features = np.hstack([tfidf_matrix.toarray(), normalized_time.values[:, np.newaxis]])

# Split the data
X_train, X_test, y_train, y_test = train_test_split(features, data['TranslatedRecipeName'], test_size=0.2, random_state=42)
```

3. Model Training:

We have trained using different regression models

KNN Training

```
from sklearn.neighbors import NearestNeighbors
import numpy as np

import joblib
# Combine TF-IDF features with normalized time
# Ensure both are numpy arrays and reshape the normalized time to match dimensions
# Ensure normalized_time is a NumPy array
normalized_time_array = normalized_time.values.reshape(-1, 1)

# Combine TF-IDF features with normalized time
combined_features = np.hstack([tfidf_matrix.toarray(), normalized_time_array])


# Use NearestNeighbors for retrieval
knn_retriever = NearestNeighbors(n_neighbors=5, metric='cosine')
knn_retriever.fit(combined_features)

# Saving the model
joblib.dump(knn_retriever, 'knn_retriever_model.pkl')
```

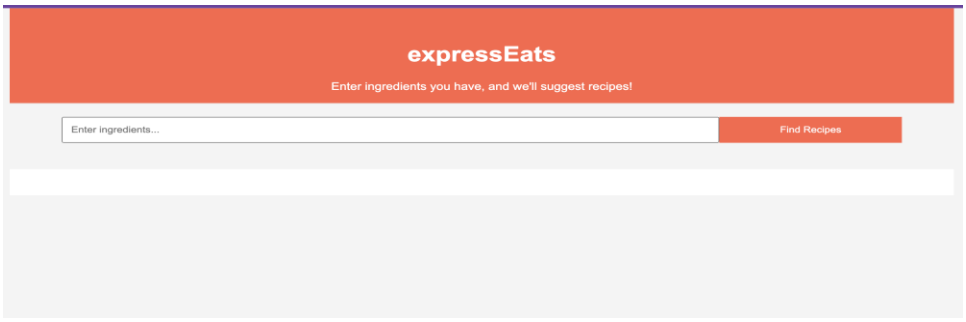
```
['knn_retriever_model.pkl']
```

4. **Recommendation Engine:**

   Implement collaborative filtering and content-based filtering algorithms.
   Developed a scoring system to prioritize recipes based on ingredients and preparation time.

5. **User interface:**
   Created a user-friendly web interface.

   The web interface has search bar and recipe recommendation functionalities.



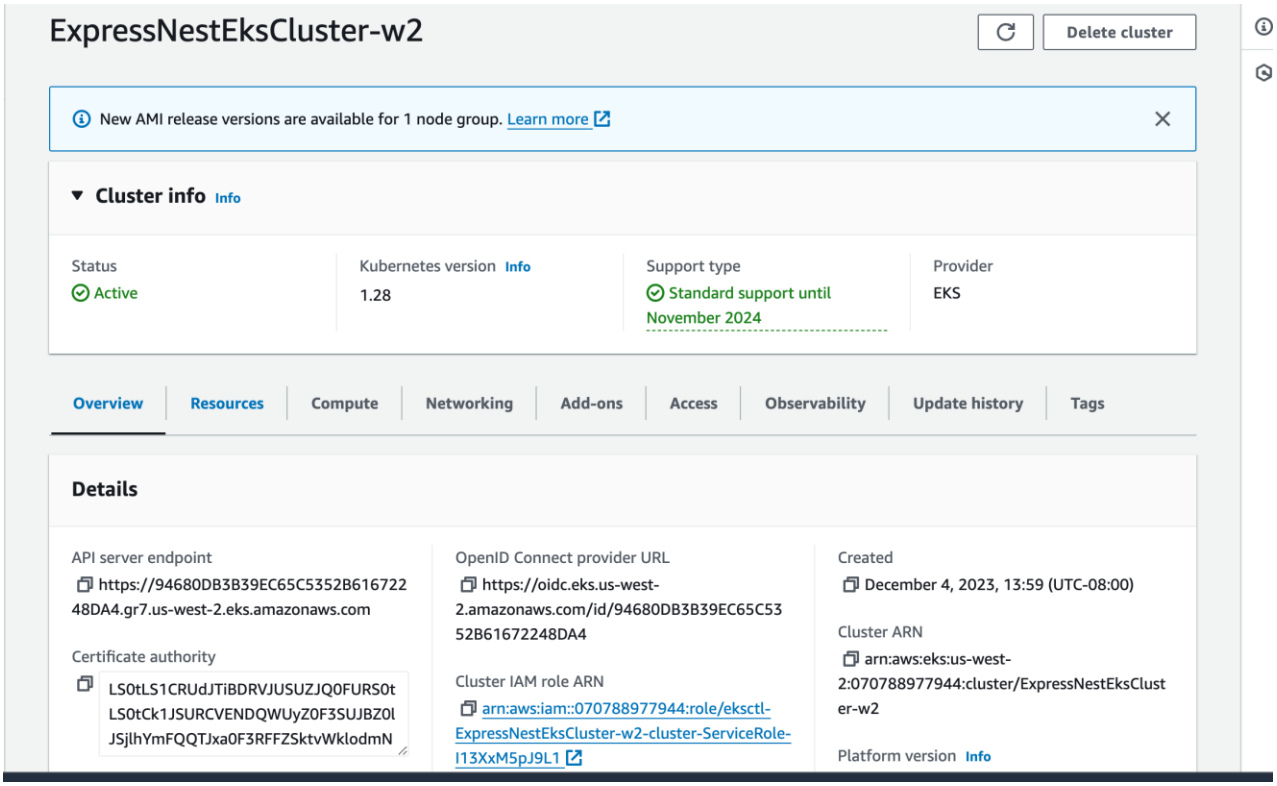
6. **Performance Evaluation:**

   We have used various metrics to evaluate model performance.

```
for query_index in range(len(data)):
    # Getting recommendations (excluding the query itself)
    recommendations_indices = indices[query_index][1:]
    recommendations_indices_list.append(recommendations_indices)

    precision, recall = precision_recall_at_k(query_index, recommendations_indices)
    precisions.append(precision)
    recalls.append(recall)
    f1_scores.append(f1_score_at_k(precision, recall))

# Calculating average metrics
average_precision = sum(precisions) / len(precisions)
average_recall = sum(recalls) / len(recalls)
average_f1_score = sum(f1_scores) / len(f1_scores)

# MRR and Coverage
mrr = mean_reciprocal_rank(indices, recommendations_indices_list)
cov = coverage(recommendations_indices_list, len(data))

# Print metrics
print(f"Average Precision@5: {average_precision}")
print(f"Average Recall@5: {average_recall}")
print(f"Average F1 Score@5: {average_f1_score}")
print(f"Mean Reciprocal Rank: {mrr}")
print(f"Coverage: {cov}")
```

```
Average Precision@5: 1.0
Average Recall@5: 0.8333333333332906
Average F1 Score@5: 0.9090909090909766
Mean Reciprocal Rank: 1.0
Coverage: 0.8665695557174071
```

7. **Deployment:**
   Deployed the Food Recipe Suggestion System on AWS platform ensuring scalability and reliability.



Nodes (2) Info

Filter Nodes by property or value

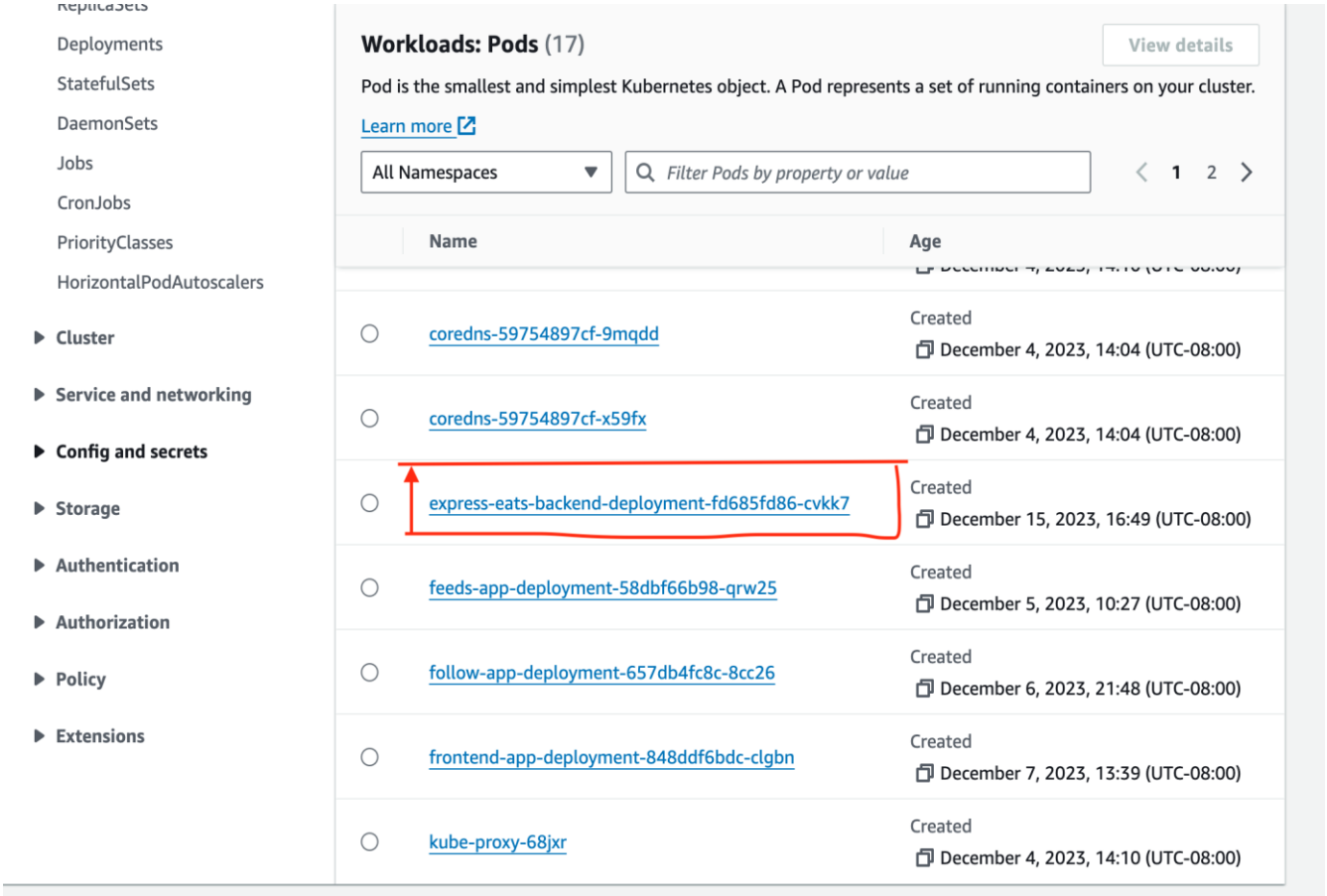
1

Node name	Instance type	Node group	Created	Status
ip-192-168-14-81.us-west-2.compute.internal	t3.medium	ExpressnestNodeGroup	Created December 4, 2023, 14:10 (UTC-08:00)	Ready
ip-192-168-65-94.us-west-2.compute.internal	t3.medium	ExpressnestNodeGroup	Created December 4, 2023, 14:10 (UTC-08:00)	Ready

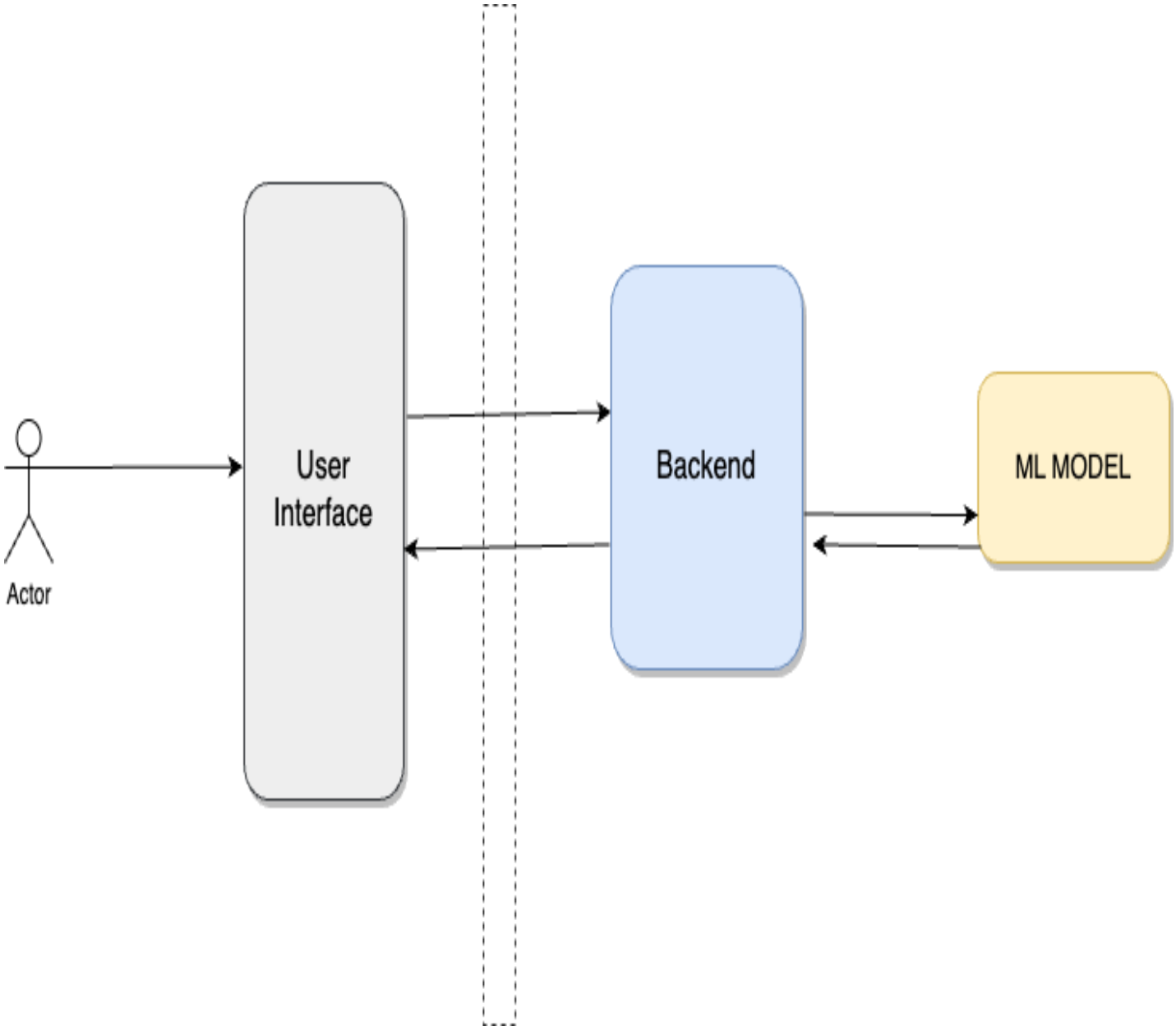
Node groups (1) Info

Edit | Delete | Add node group

Group name	Desired size	AMI release version	Launch template
ExpressnestNodeGroup	2	1.28.3-20231116 Update now	eksctl-ExpressNestEksCluster-w2-nodegroup-ExpressnestNod



Architecture:


Actor
User Interface
Backend
ML MODEL

Experiment and Results:

Phase 1: Business Understanding

Our objective is to develop a system that, given the total time and translated ingredients, recommends a recipe that is closest to these criteria. This can be viewed as a recommendation system problem where the input features are time and ingredients, and the output is a recipe recommendation.

Phase 2: Data Understanding

Started by loading the dataset and performed an initial analysis to understand its structure, content, and quality. This will include looking at the number of records, features, any missing values, and the types of data we are dealing with.

We did EDA using Sweetviz

Phase 3: Data Preparation

This phase involves preparing the dataset for modeling. Since our goal is to recommend a recipe based on total time and translated ingredients, we will focus on these aspects.

In this phase we have handled missing values followed by feature engineering and feature selection.

We split the data after that.

Splitting the Data

```
from sklearn.neighbors import NearestNeighbors
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
data.dropna(subset=['TotalTimeInMins', 'TranslatedIngredients', 'TranslatedRecipeName'], inplace=True)

# Vectorize the ingredients
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(data['TranslatedIngredients'])
joblib.dump(tfidf, 'tfidf_vectorizer.pkl')
# Normalize TotalTimeInMins
max_time = data['TotalTimeInMins'].max()
normalized_time = data['TotalTimeInMins'] / max_time

# Combine time with ingredient vectors
features = np.hstack([tfidf_matrix.toarray(), normalized_time.values[:, np.newaxis]])

# Split the data
X_train, X_test, y_train, y_test = train_test_split(features, data['TranslatedRecipeName'], test_size=0.2, random_state=42)
```

Phase 4: Modeling

We will create functions to:

Train Models: Train different regression models.

Inference: Predict using the models.

Evaluation: Use various metrics to evaluate model performance.

Saving Models: Save each trained model and its vectorizer.

KNN Training

```
from sklearn.neighbors import NearestNeighbors
import numpy as np

import joblib
# Combine TF-IDF features with normalized time
# Ensure both are numpy arrays and reshape the normalized time to match dimensions
# Ensure normalized_time is a NumPy array
normalized_time_array = normalized_time.values.reshape(-1, 1)

# Combine TF-IDF features with normalized time
combined_features = np.hstack([tfidf_matrix.toarray(), normalized_time_array])


# Use NearestNeighbors for retrieval
knn_retriever = NearestNeighbors(n_neighbors=5, metric='cosine')
knn_retriever.fit(combined_features)

# Saving the model
joblib.dump(knn_retriever, 'knn_retriever_model.pkl')
```

```
['knn_retriever_model.pkl']
```

Evaluation

KNN Inference

```
def make_recommendations(input_ingredients, input_time, model, tfidf_vectorizer, data, num_recommendations=5):
    # Process the input
    input_vector = tfidf_vectorizer.transform([input_ingredients])

    # Convert and reshape normalized time
    normalized_time = np.array([input_time / max_time]).reshape(-1, 1)
    input_features = np.hstack([input_vector.toarray(), normalized_time])

    # Find nearest recipes
    distances, indices = model.kneighbors(input_features)

    # Get recommended recipe names
    recommended_recipes = data.iloc[indices[0]]['TranslatedRecipeName']
    return recommended_recipes
```

```
[ ] # Example input
    input_ingredients = "1 cup rice, 2 tomatoes, 1 tsp salt"
    input_time = 30  # in minutes

    # Load models and vectorizer
    knn_retriever = joblib.load('knn_retriever_model.pkl')
    tfidf_vectorizer = joblib.load('tfidf_vectorizer.pkl')

    # Get recommendations
    recommendations = make_recommendations(input_ingredients, input_time, knn_retriever, tfidf_vectorizer, data)
    print(recommendations)
```

```
607                    Madurai Spicy Tomato Chutney Recipe
5388      Potato Bonda Recipe - Low Fat Aloo Bonda Recipe
5432                                 Patiala Aloo Recipe
413            Mullu Murukku Recipe - Mullu Murukku Recipe
1038    Tomato Onion Chutney Recipe - Tomato Onion Chu...
Name: TranslatedRecipeName, dtype: object
```

KNN All possible Metrics!!

Precision@k: The proportion of recommended items in the top-k set that are relevant.

Recall@k: The proportion of relevant items found in the top-k recommendations.

F1 Score@k: The harmonic means of precision and recall at k.

Mean Reciprocal Rank (MRR): The average of the reciprocal ranks of the first relevant recommendation.

Coverage: The proportion of items in the dataset for which the recommender can provide recommendations.

```
for query_index in range(len(data)):
    # Getting recommendations (excluding the query itself)
    recommendations_indices = indices[query_index][1:]
    recommendations_indices_list.append(recommendations_indices)

    precision, recall = precision_recall_at_k(query_index, recommendations_indices)
    precisions.append(precision)
    recalls.append(recall)
    f1_scores.append(f1_score_at_k(precision, recall))

# Calculating average metrics
average_precision = sum(precisions) / len(precisions)
average_recall = sum(recalls) / len(recalls)
average_f1_score = sum(f1_scores) / len(f1_scores)

# MRR and Coverage
mrr = mean_reciprocal_rank(indices, recommendations_indices_list)
cov = coverage(recommendations_indices_list, len(data))

# Print metrics
print(f"Average Precision@5: {average_precision}")
print(f"Average Recall@5: {average_recall}")
print(f"Average F1 Score@5: {average_f1_score}")
print(f"Mean Reciprocal Rank: {mrr}")
print(f"Coverage: {cov}")
```

```
Average Precision@5: 1.0
Average Recall@5: 0.8333333333332906
Average F1 Score@5: 0.9090909090909766
Mean Reciprocal Rank: 1.0
Coverage: 0.8665695557174071
```

Data Loading, Cleaning, Tokenization

KMeans Training

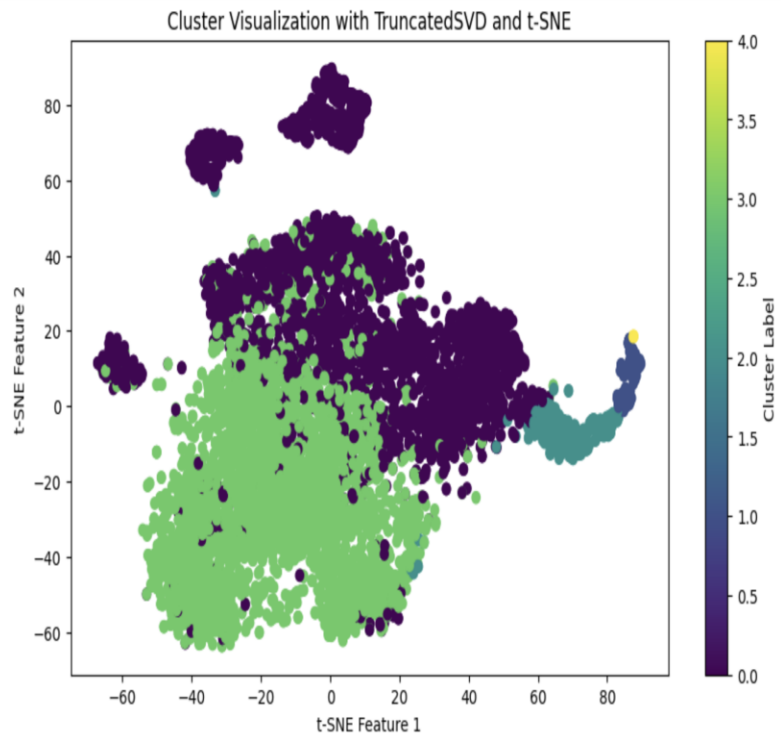
```
# Step 4: Train KMeans Model
n_clusters = 5
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
kmeans.fit(X_combined)
data_cleaned['Cluster'] = kmeans.labels_
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the
  warnings.warn(
/tmp/ipykernel_19/3370755713.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_cleaned['Cluster'] = kmeans.labels_
```

KMeans Inference

```
/opt/conda/lib/python3.10/site-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
Srno                                                         1751
RecipeName              Pudina Mavinakayi Chutney Recipe - Mint Mango ...
TranslatedRecipeName    Pudina Mavinakayi Chutney Recipe - Mint Mango ...
Ingredients             1 Mango (Raw) - grated,1 cup Fresh coconut,1/2...
TranslatedIngredients   1 Mango (Raw) - grated,1 cup Fresh coconut,1/2...
PrepTimeInMins                                                        15
CookTimeInMins                                                         0
TotalTimeInMins                                                       15
Servings                                                               4
Cuisine                                                        Karnataka
Course                                                         Side Dish
Diet                                                         Vegetarian
Instructions            To begin making the Karnataka Style Pudina Mav...
TranslatedInstructions  To begin making the Karnataka Style Pudina Mav...
URL                     http://www.archanaskitchen.com/karnataka-style...
Cluster                                                                3
Name: 1552, dtype: object
/tmp/ipykernel_19/2735290301.py:28: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cluster_data['Distance'] = cluster_data.index.map(
TranslatedRecipeName    Pudina Mavinakayi Chutney Recipe - Mint Mango ...
Distance                                                              0.0
Name: 1552, dtype: object
```

KMeans Cluster Visualization and cluster metrics

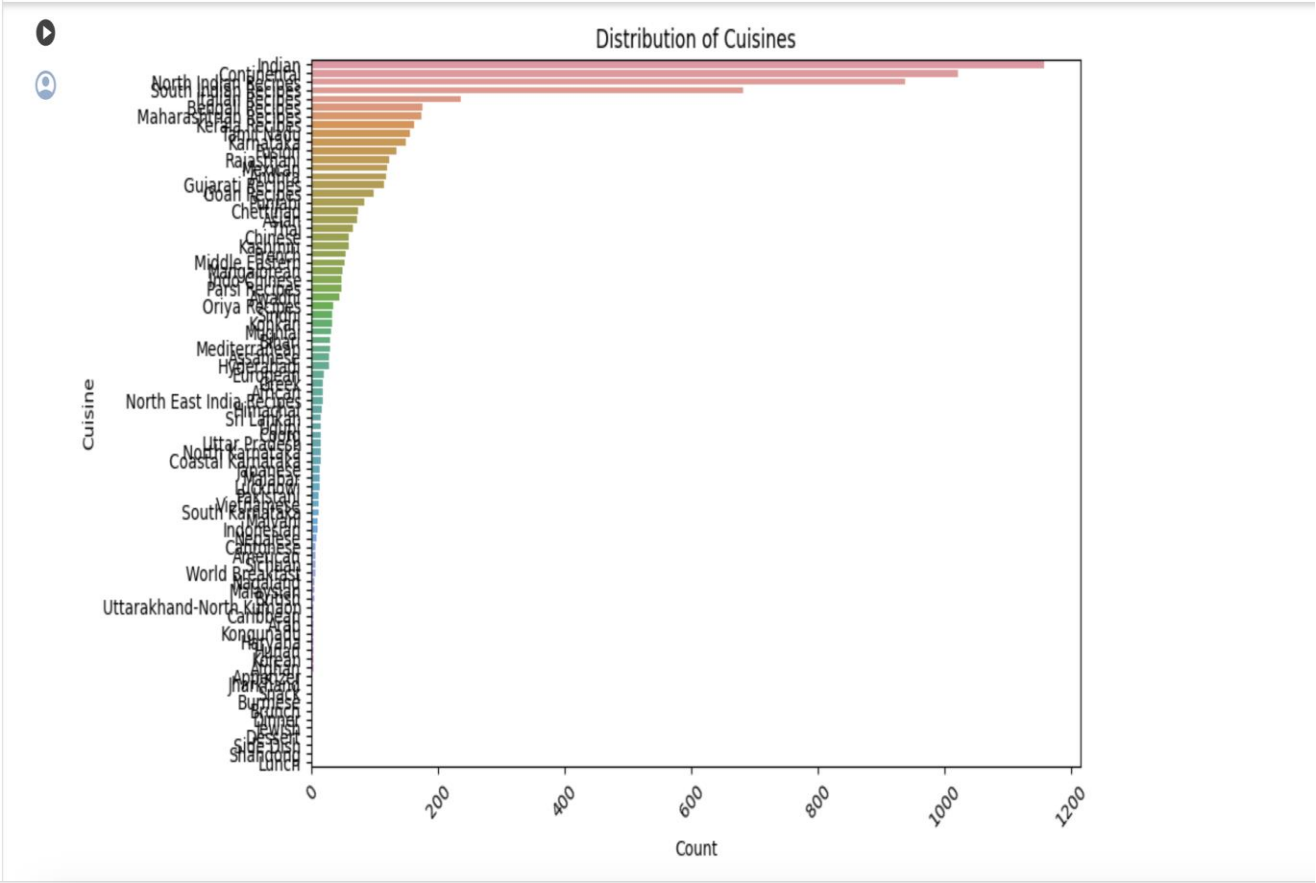


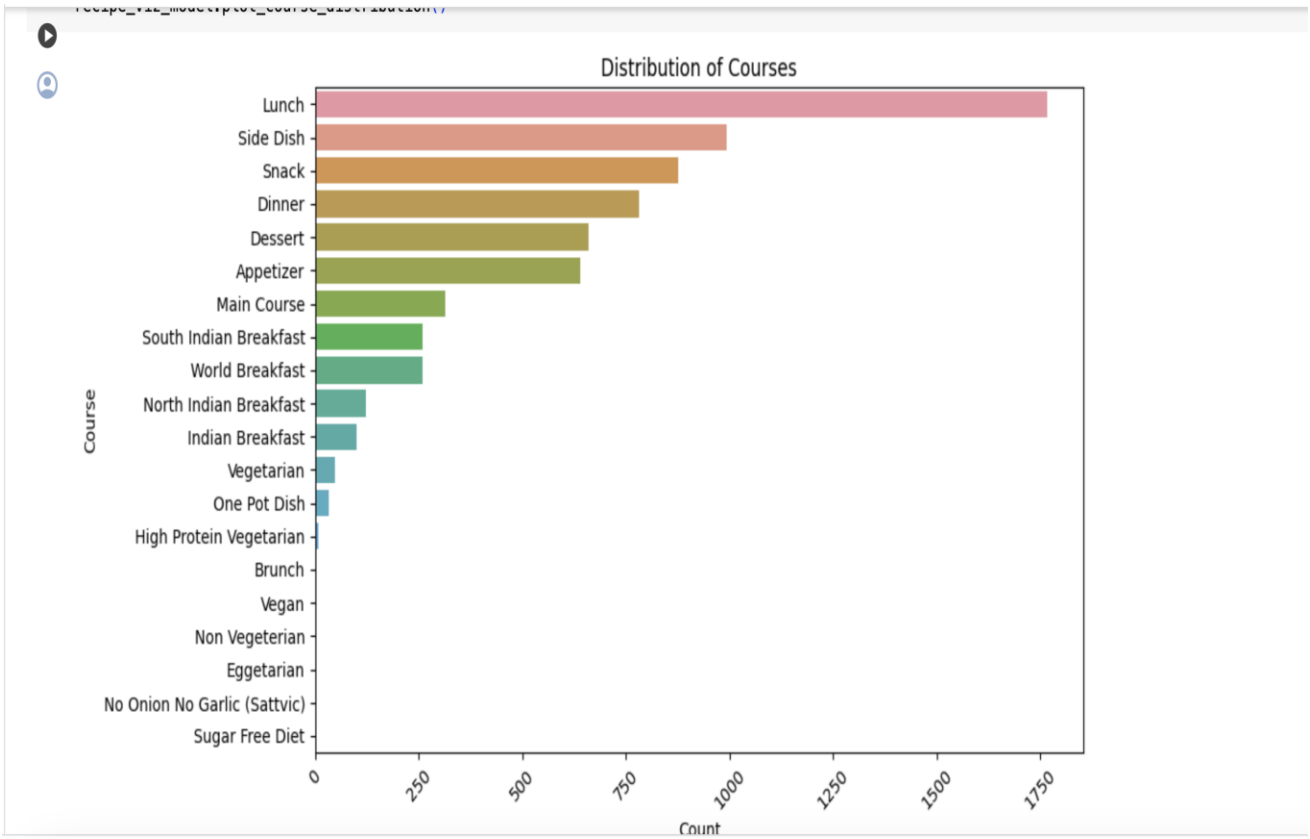

```
(0.09787780826754199, 3307.8025971898037, 1.727193328699558)
```

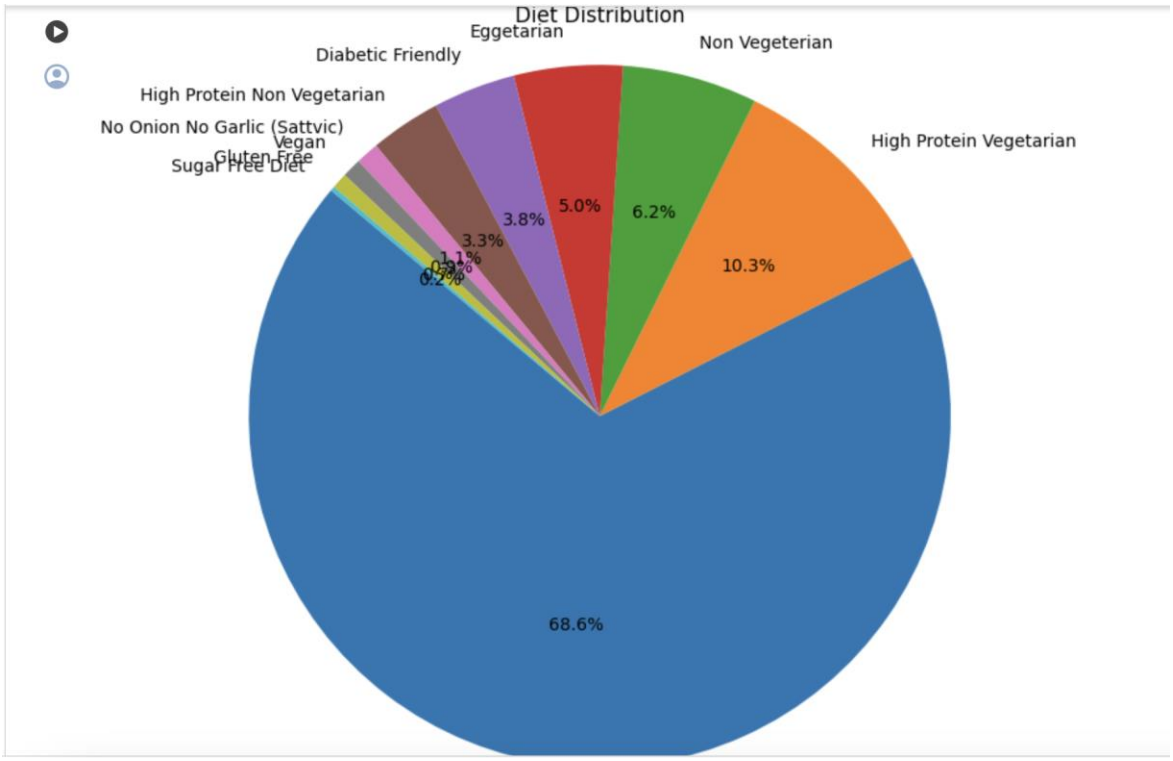
All Possible Task Metrics:

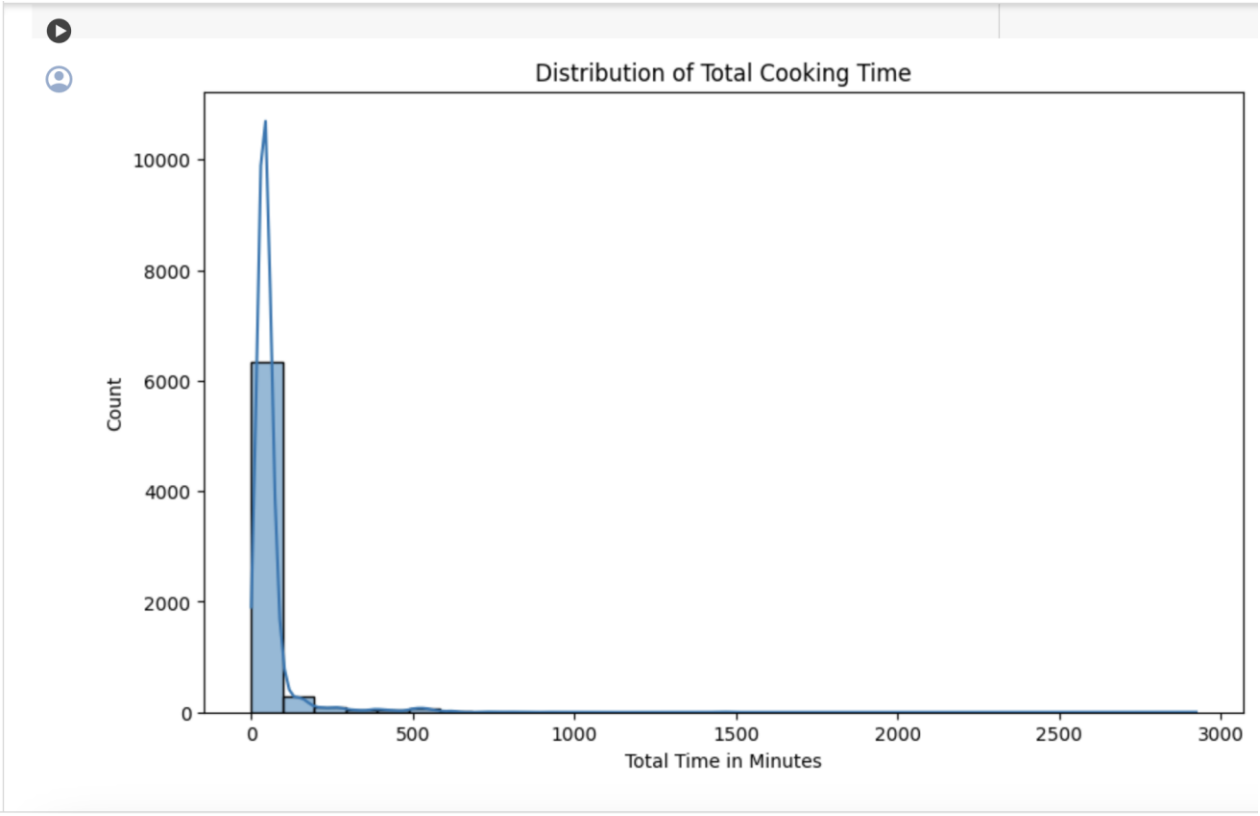
(0.10450198304965984,
 0.9978160525203789,
 0.18918996667461824,
 9.734318895179761e-07,
 0.00025874879933965853)

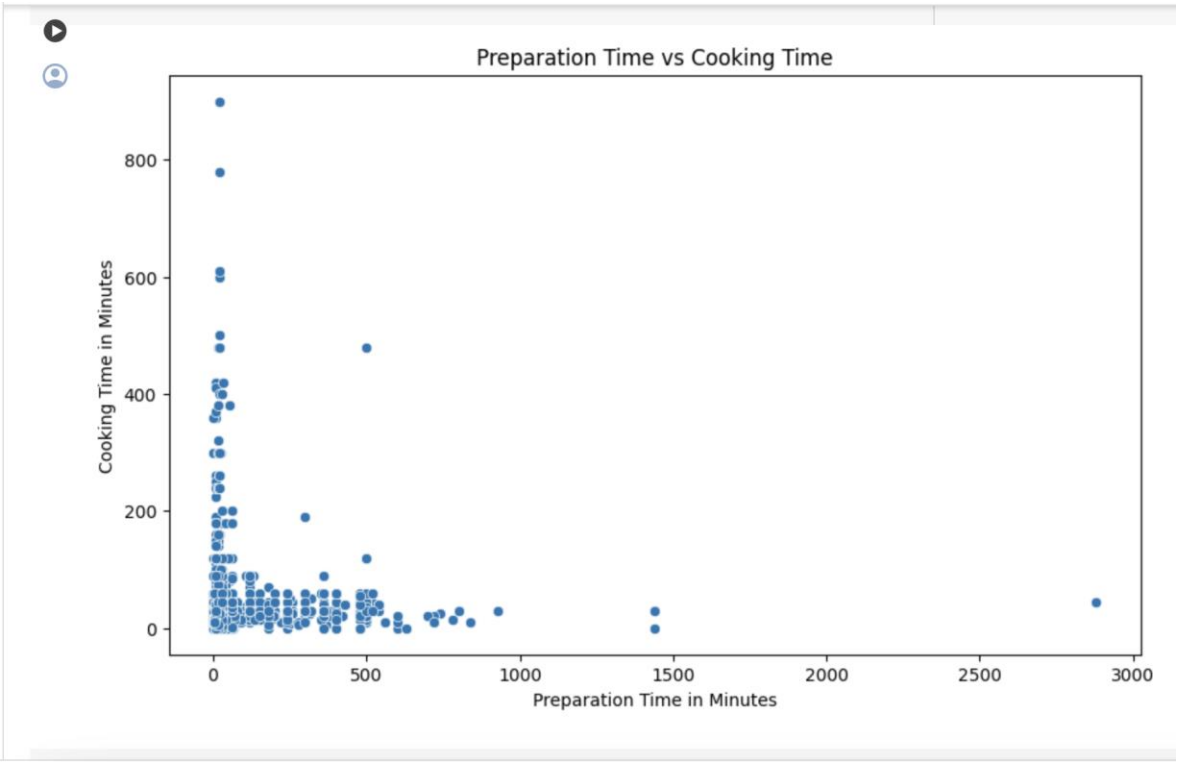
Visualizations

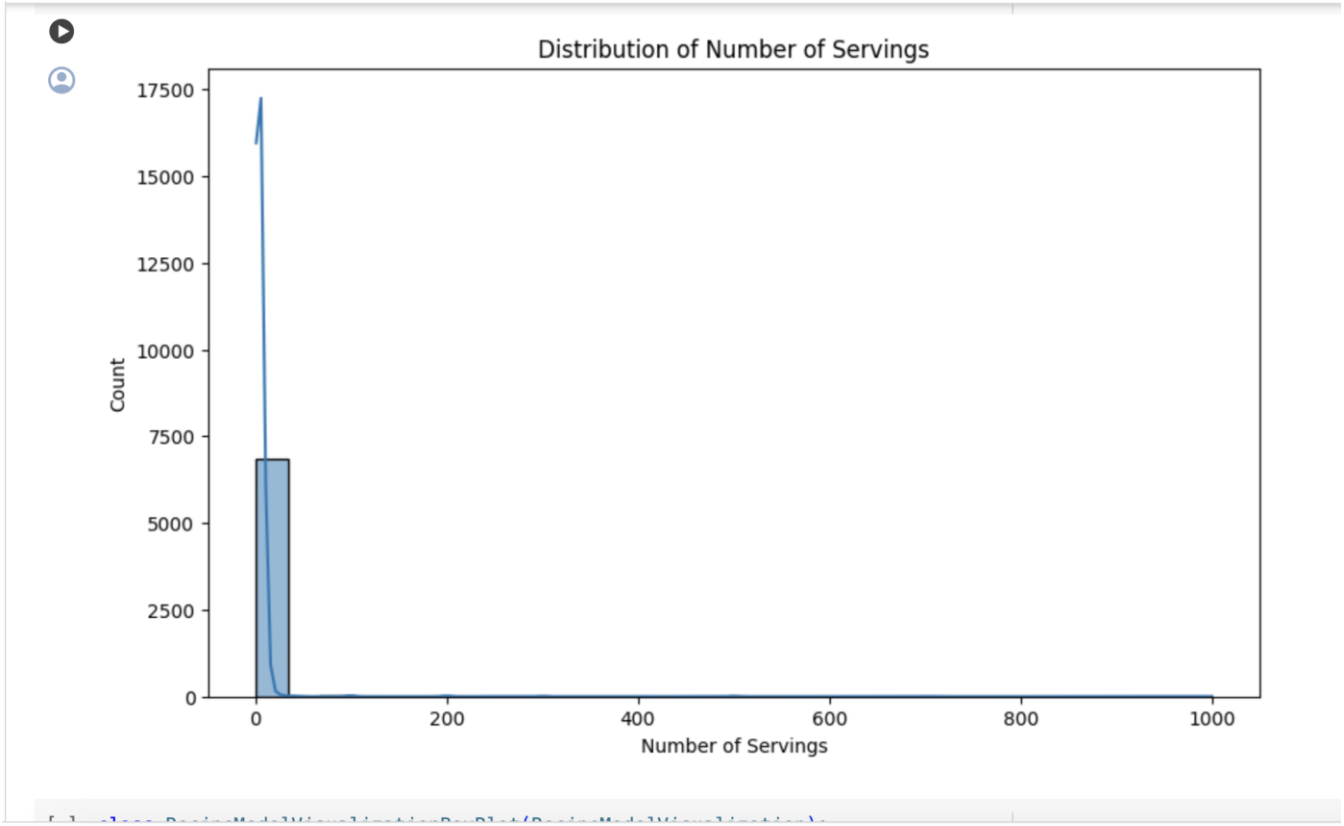


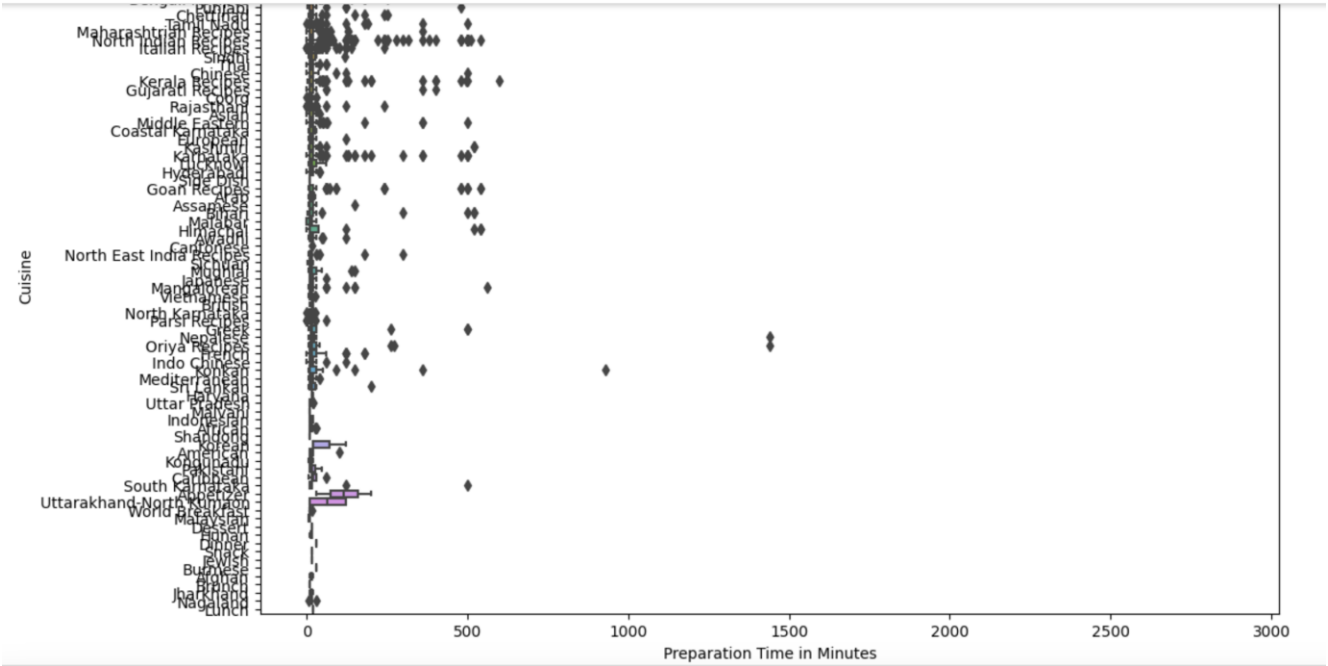

Distribution of Courses
Lunch
Side Dish
Snack
Dinner
Dessert
Appetizer
Main Course
South Indian Breakfast
World Breakfast
North Indian Breakfast
Indian Breakfast
Vegetarian
One Pot Dish
High Protein Vegetarian
Brunch
Vegan
Non Vegeterian
Eggetarian
No Onion No Garlic (Sattvic)
Sugar Free Diet
Course
0
250
500
750
1000
1250
1500
1750
Count

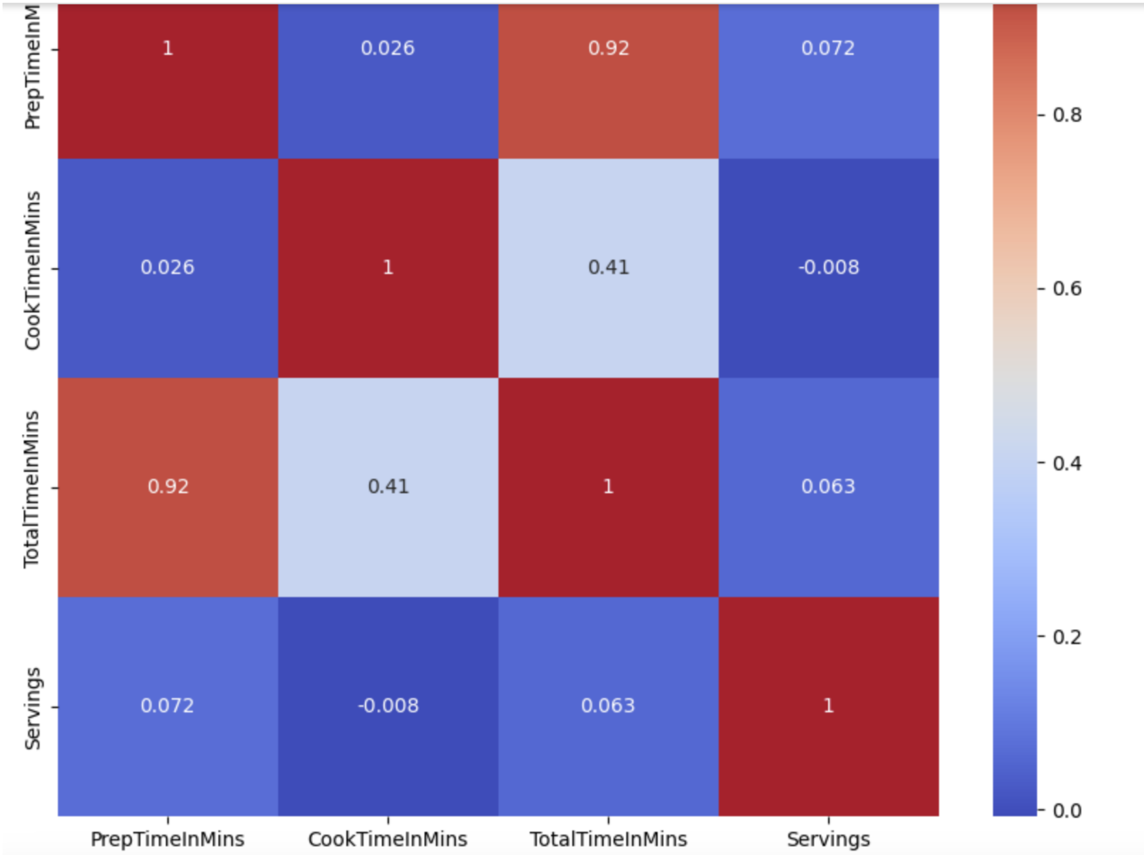

Diet Distribution
Eggetarian
Non Vegeterian
Diabetic Friendly
High Protein Non Vegetarian
No Onion No Garlic (Sattvic)
Vegan
Gluten Free
Sugar Free Diet
High Protein Vegetarian
5.0%
6.2%
3.8%
3.3%
10.3%
68.6%

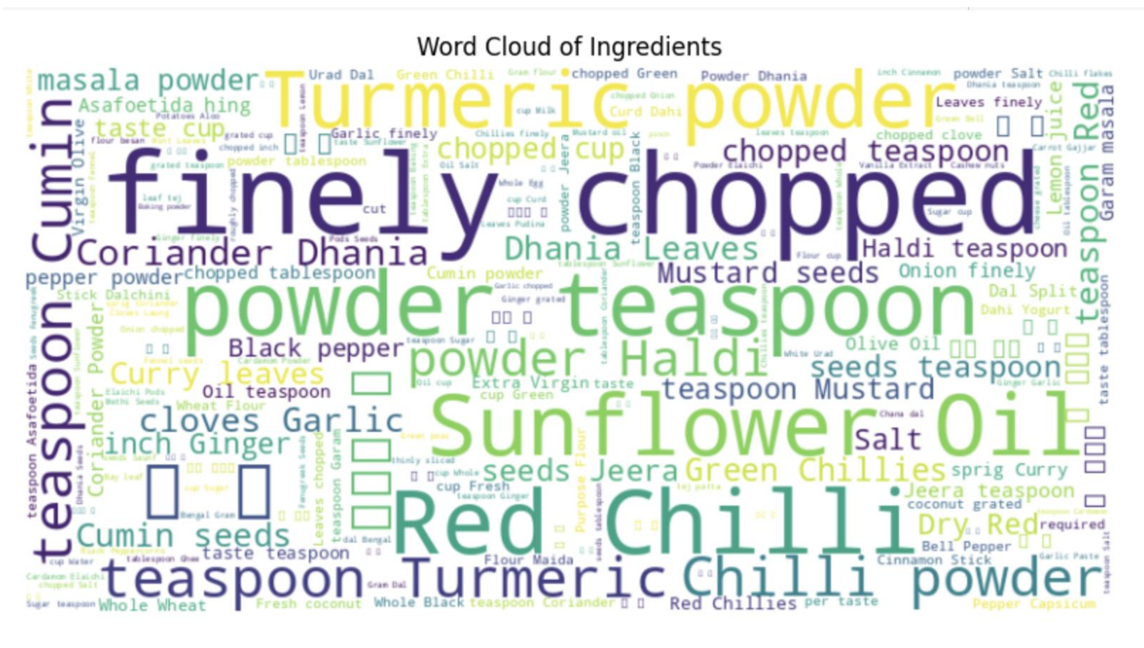

Distribution of Total Cooking Time
Count
Total Time in Minutes
0
2000
4000
6000
8000
10000
0
500
1000
1500
2000
2500
3000

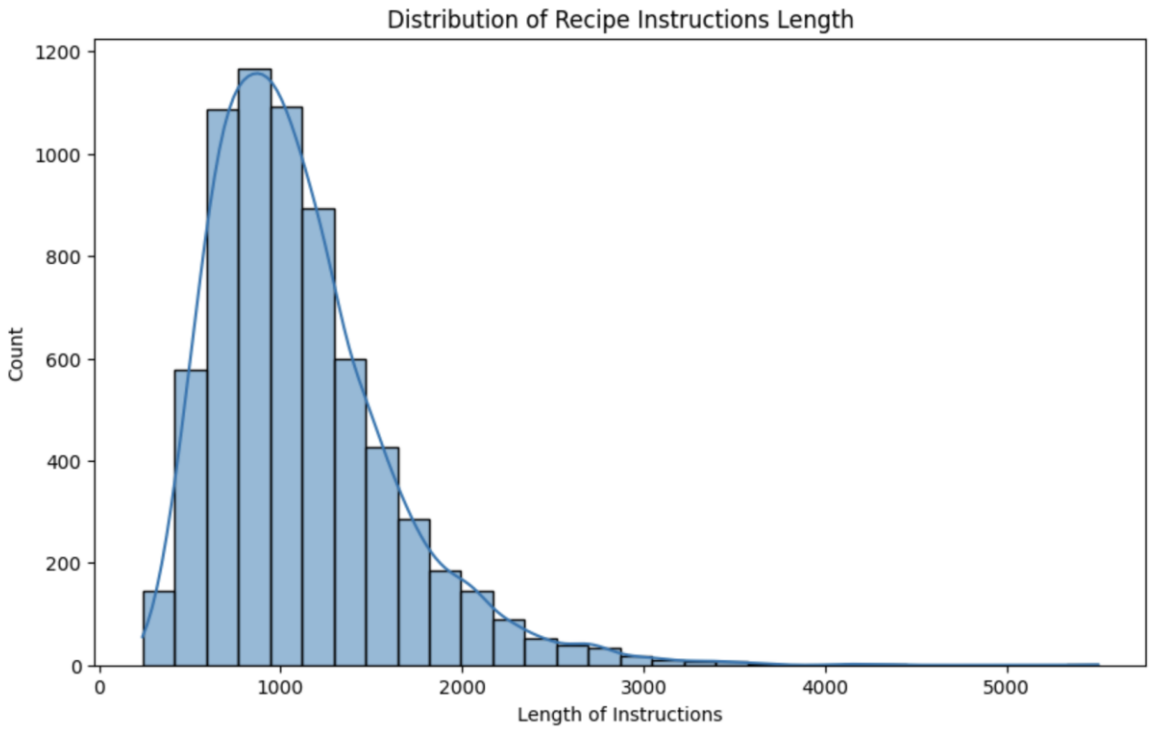

Preparation Time vs Cooking Time
Cooking Time in Minutes
Preparation Time in Minutes
0
200
400
600
800
0
500
1000
1500
2000
2500
3000


Distribution of Number of Servings
Count
Number of Servings
0
2500
5000
7500
10000
12500
15000
17500
0
200
400
600
800
1000


Cuisine
Preparation Time in Minutes
0
500
1000
1500
2000
2500
3000


PrepTimeInMins
CookTimeInMins
TotalTimeInMins
Servings
1
0.026
0.92
0.072
0.026
1
0.41
-0.008
0.92
0.41
1
0.063
0.072
-0.008
0.063
1
0.8
0.6
0.4
0.2
0.0


Word Cloud of Ingredients
finely chopped
powder teaspoon
Sunflower Oil
Red Chilli
Turmeric powder
teaspoon Turmeric
Chilli powder
teaspoon Cumin
Coriander Dhania
powder Haldi
Dhania Leaves
chopped teaspoon
masala powder
Mustard seeds
Haldi teaspoon
teaspoon Mustard
seeds teaspoon
cloves Garlic
inch Ginger
Green Chillies
seeds Jeera
Cumin seeds
Curry leaves
Black pepper
Salt
taste teaspoon
Asafoetida hing
taste cup
chopped cup
pepper powder
Onion finely
Olive Oil
Jeera teaspoon
Dry Red
Lemon juice
Garam masala

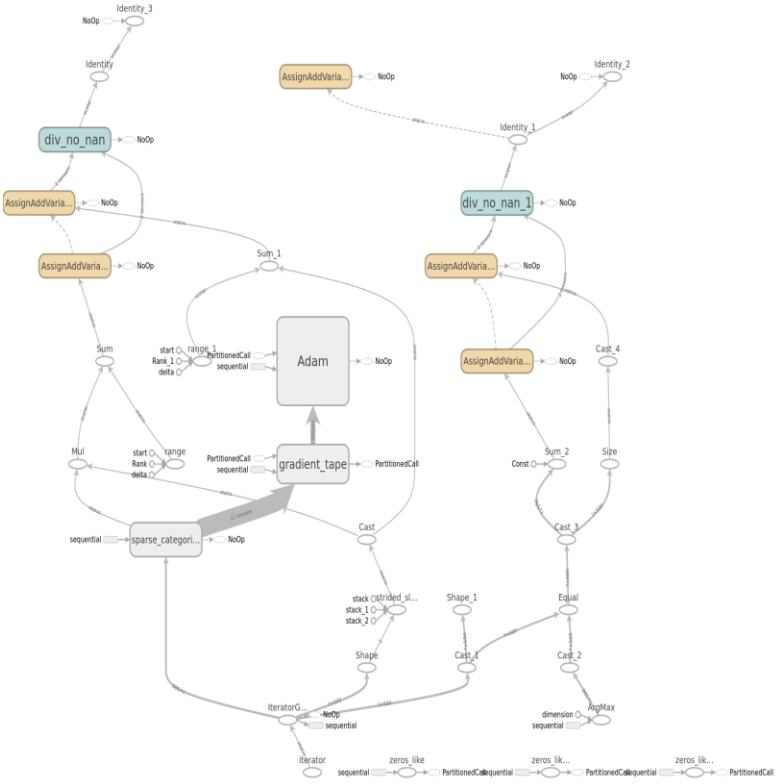

Distribution of Recipe Instructions Length
Count
Length of Instructions
0
200
400
600
800
1000
1200
0
1000
2000
3000
4000
5000

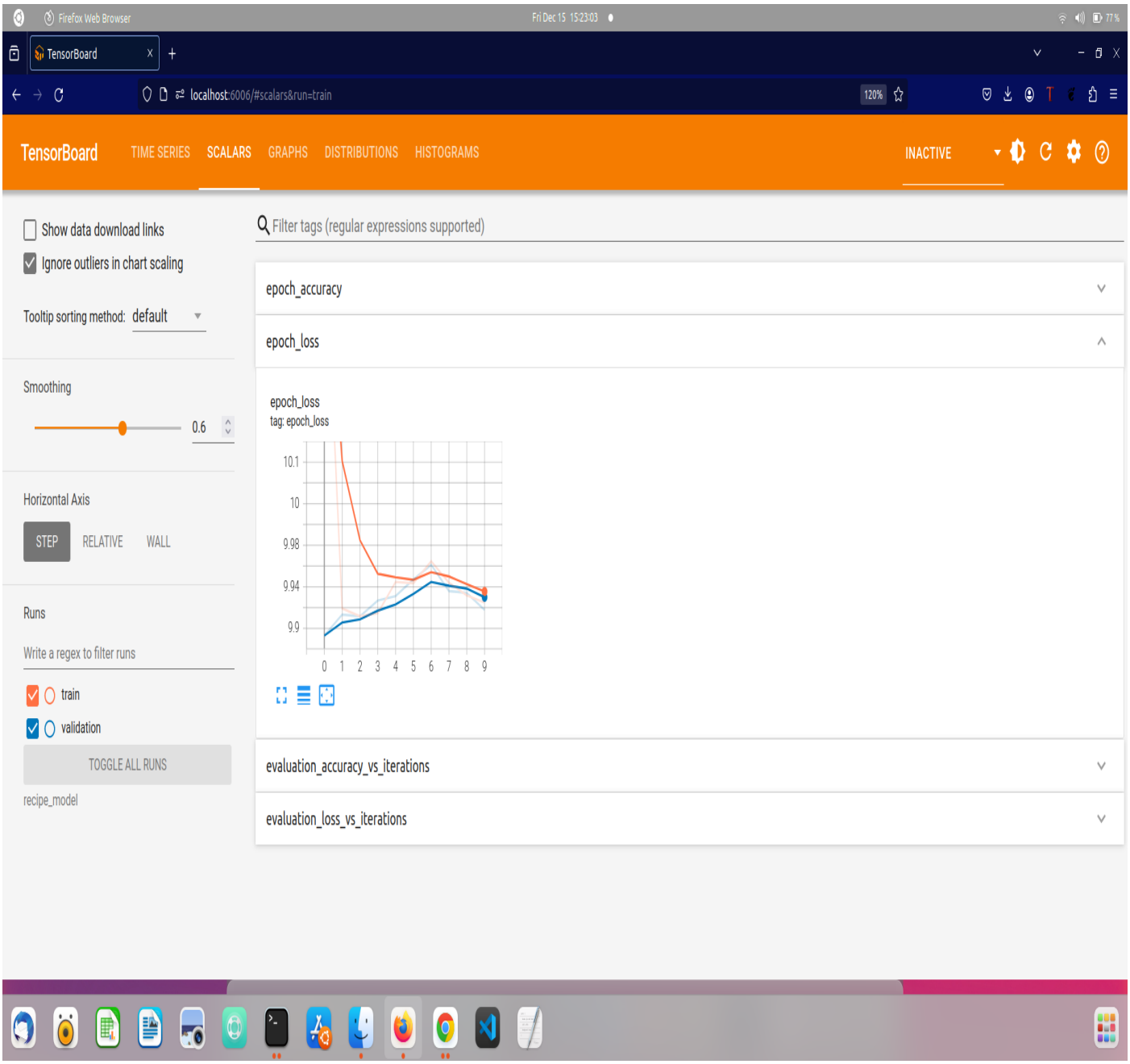
Model Training

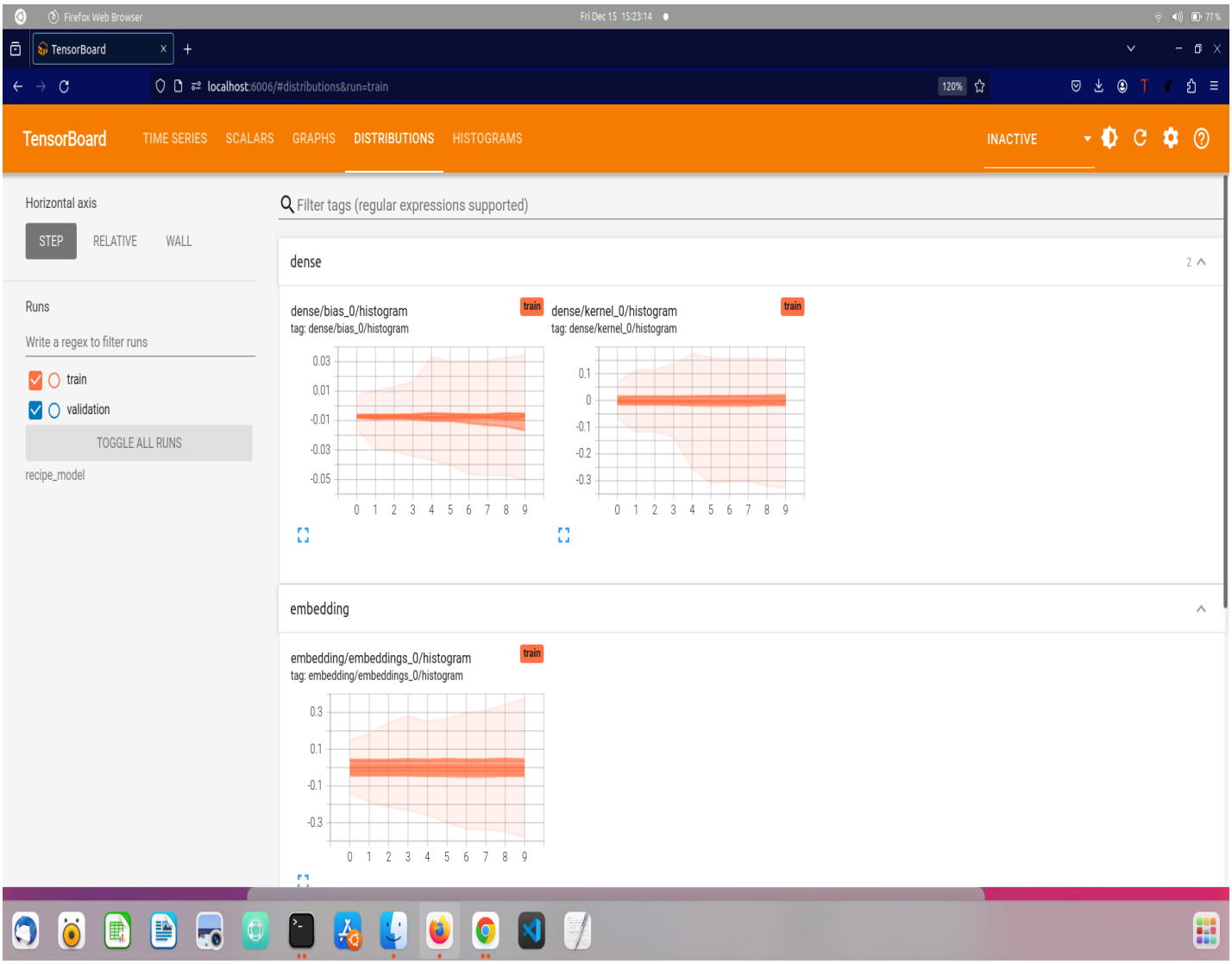
Serious Note: Accuracy does not make sense as a metric here as we are modelling sequence generation

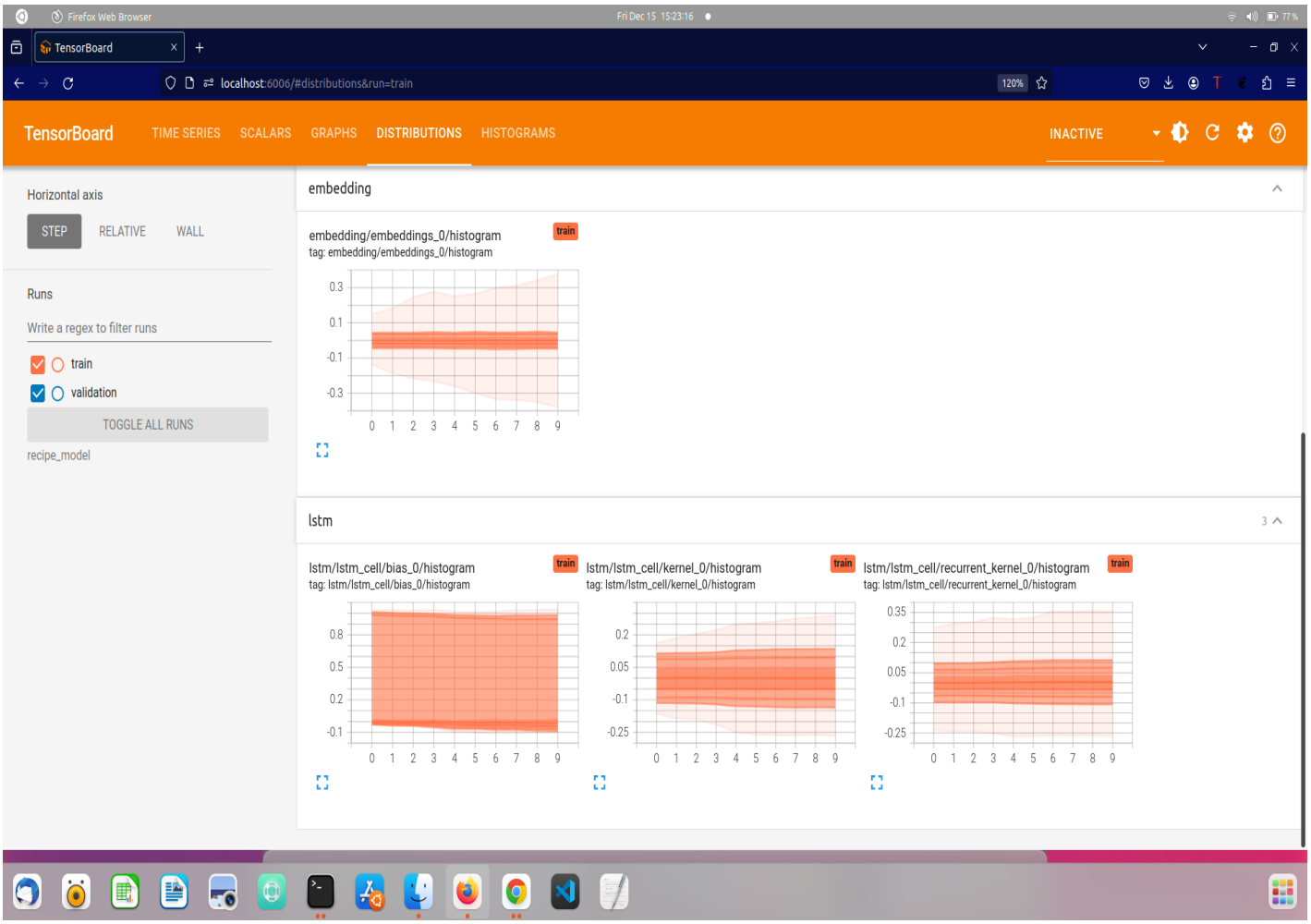
```
[ ] from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
    from tensorflow.keras.callbacks import ModelCheckpoint

    # Define a file path pattern for saving the model
    model_checkpoint = ModelCheckpoint(
        filepath="model_weights_epoch_{epoch:02d}.h5",  # Change the file path as needed
        save_weights_only=True,  # Save only the model's weights
        save_best_only=True,  # Save weights at every epoch, not just the best
        monitor="val_loss",  # You can choose a metric to monitor, e.g., validation loss
        mode="auto",  # Automatically choose the best direction to minimize the monitored metric
        verbose=1  # Display progress during training
    )

    # Define model parameters
    vocab_size = len(tokenizer.word_index) + 1  # Adding 1 because of reserved 0 index
    embedding_dim = 100  # Size of the word embeddings
    lstm_units = 128  # Number of units in LSTM layer

    # Build the model
    model = Sequential()
    model.add(Embedding(vocab_size, embedding_dim, input_length=max_length))
    model.add(LSTM(lstm_units, return_sequences=True))
    model.add(Dropout(0.2))
    model.add(Dense(vocab_size, activation='softmax'))
```

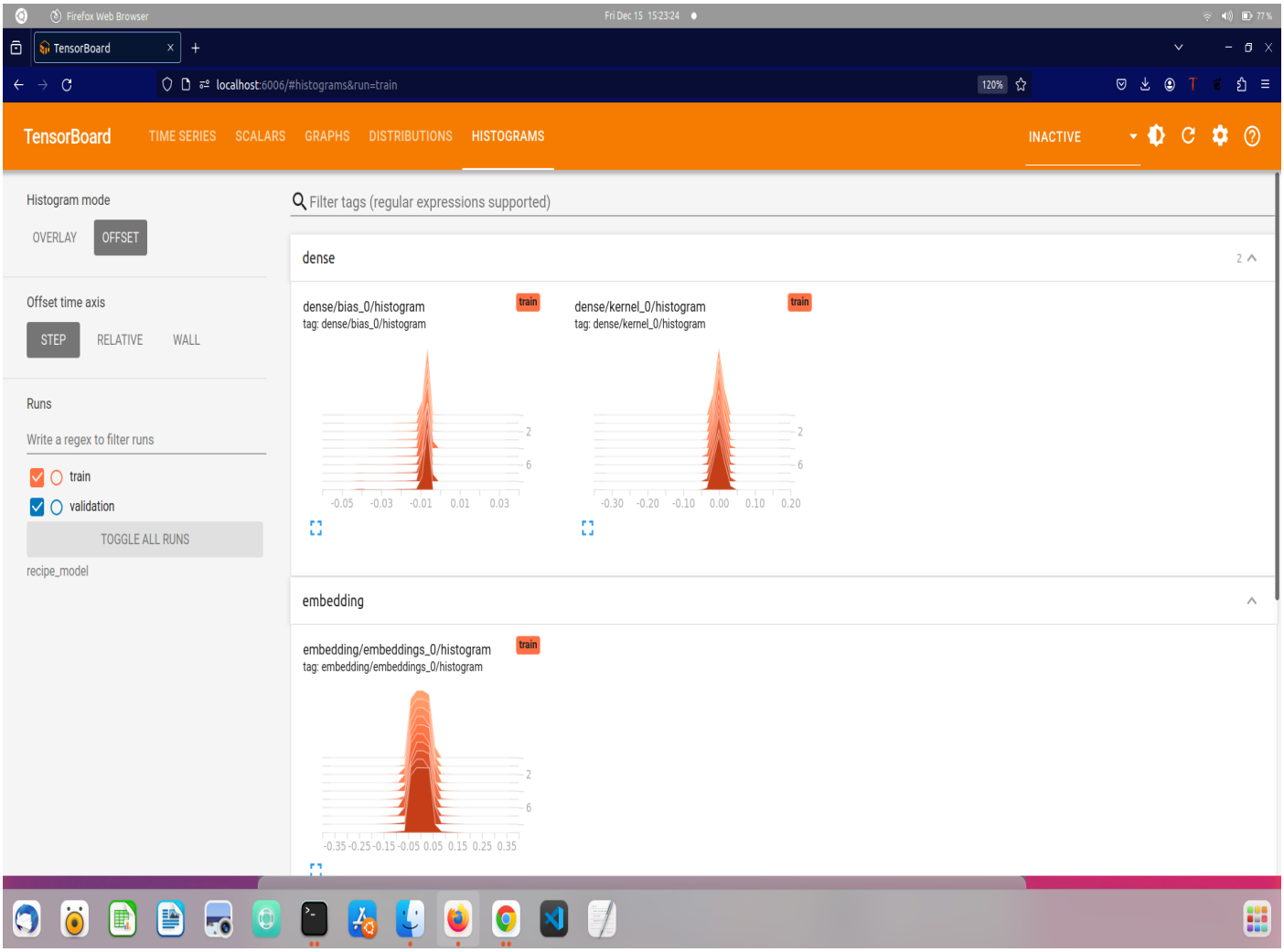
Tensor board

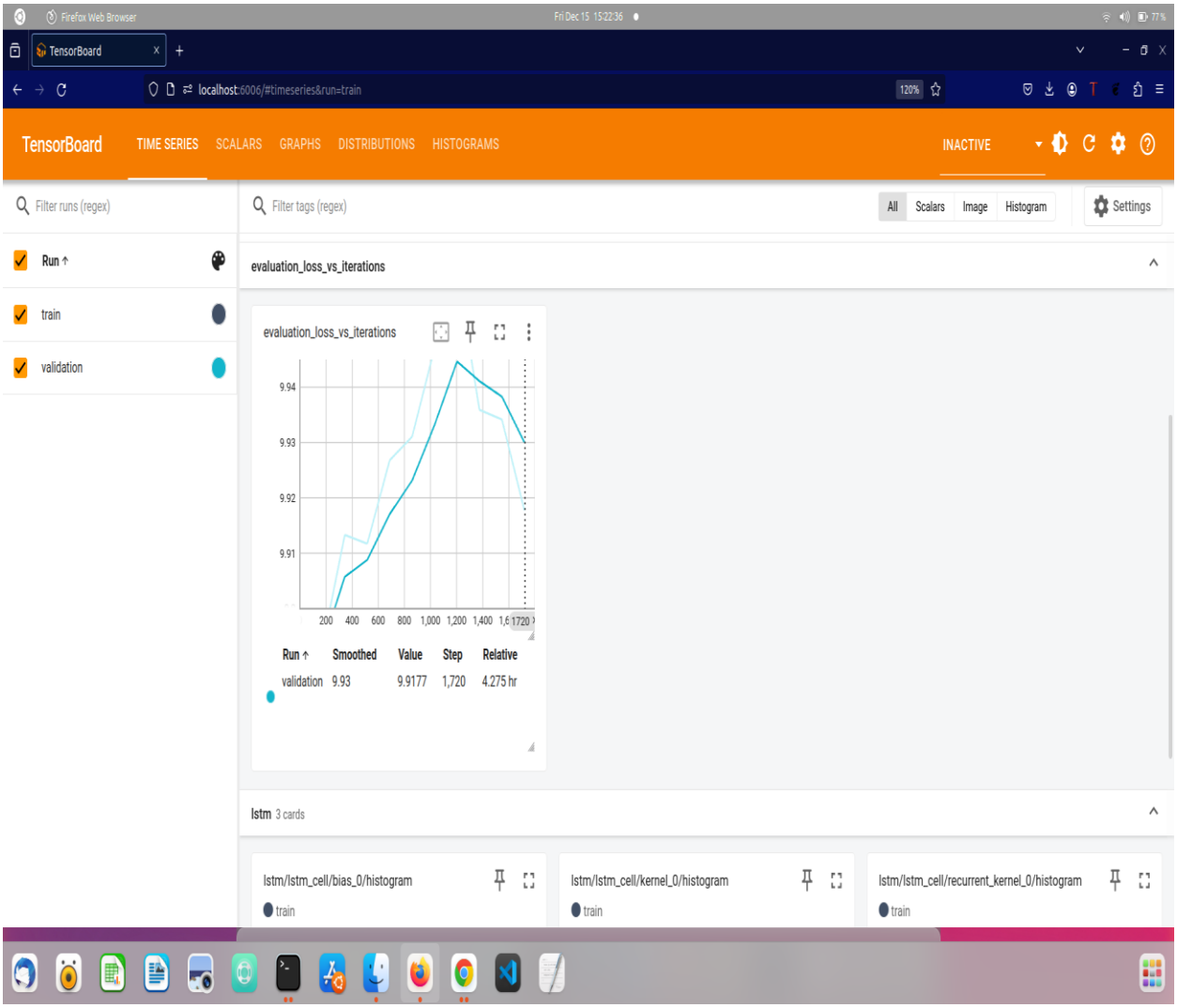


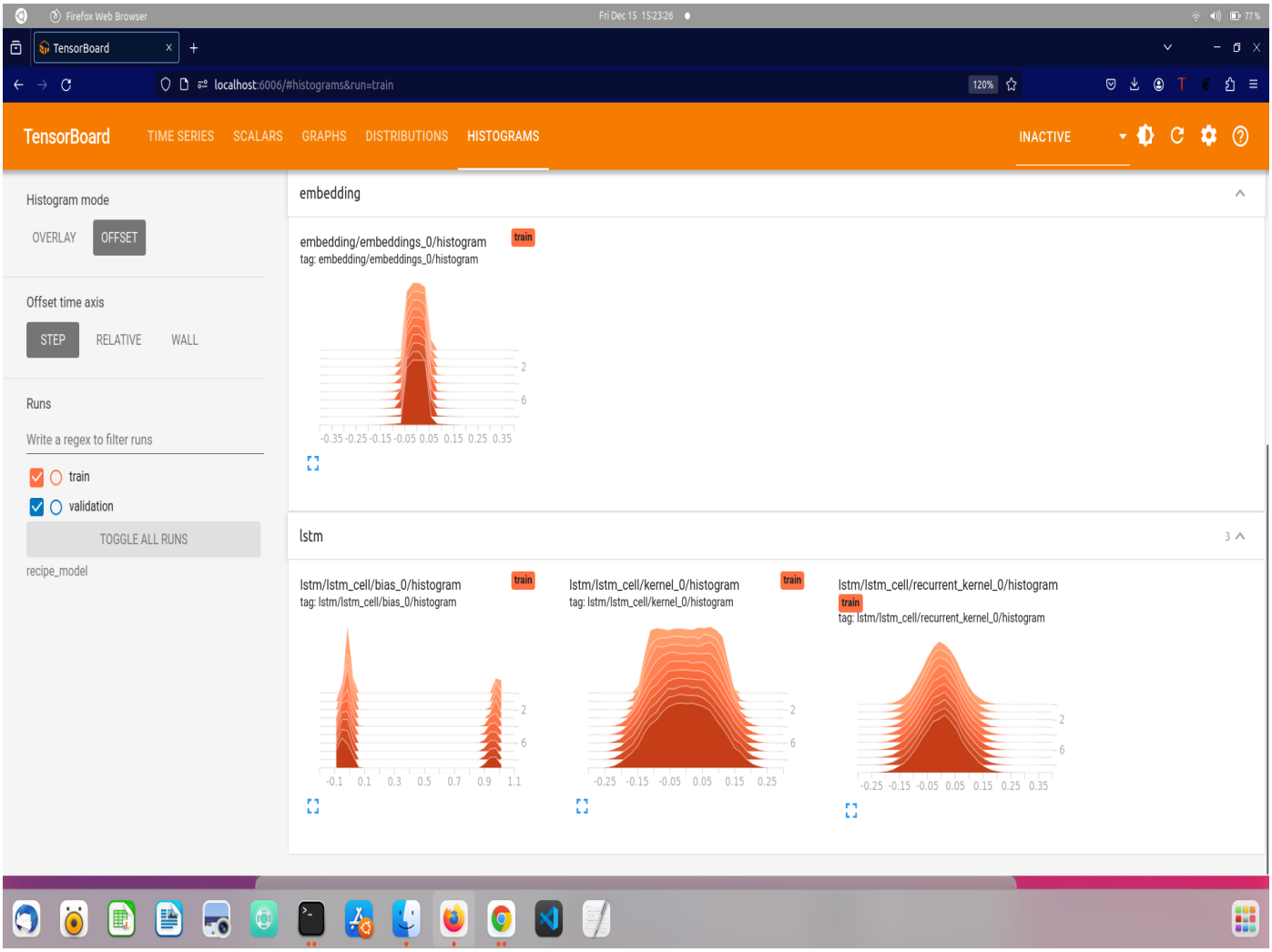

Firefox Web Browser
Fri Dec 15 15:23:03
TensorBoard
localhost:6006/#scalars&run=train
120%
TensorBoard
TIME SERIES
SCALARS
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
train
validation
TOGGLE ALL RUNS
recipe_model
Filter tags (regular expressions supported)
epoch_accuracy
epoch_loss
epoch_loss
tag: epoch_loss
10.1
10
9.98
9.94
9.9
0 1 2 3 4 5 6 7 8 9
evaluation_accuracy_vs_iterations
evaluation_loss_vs_iterations

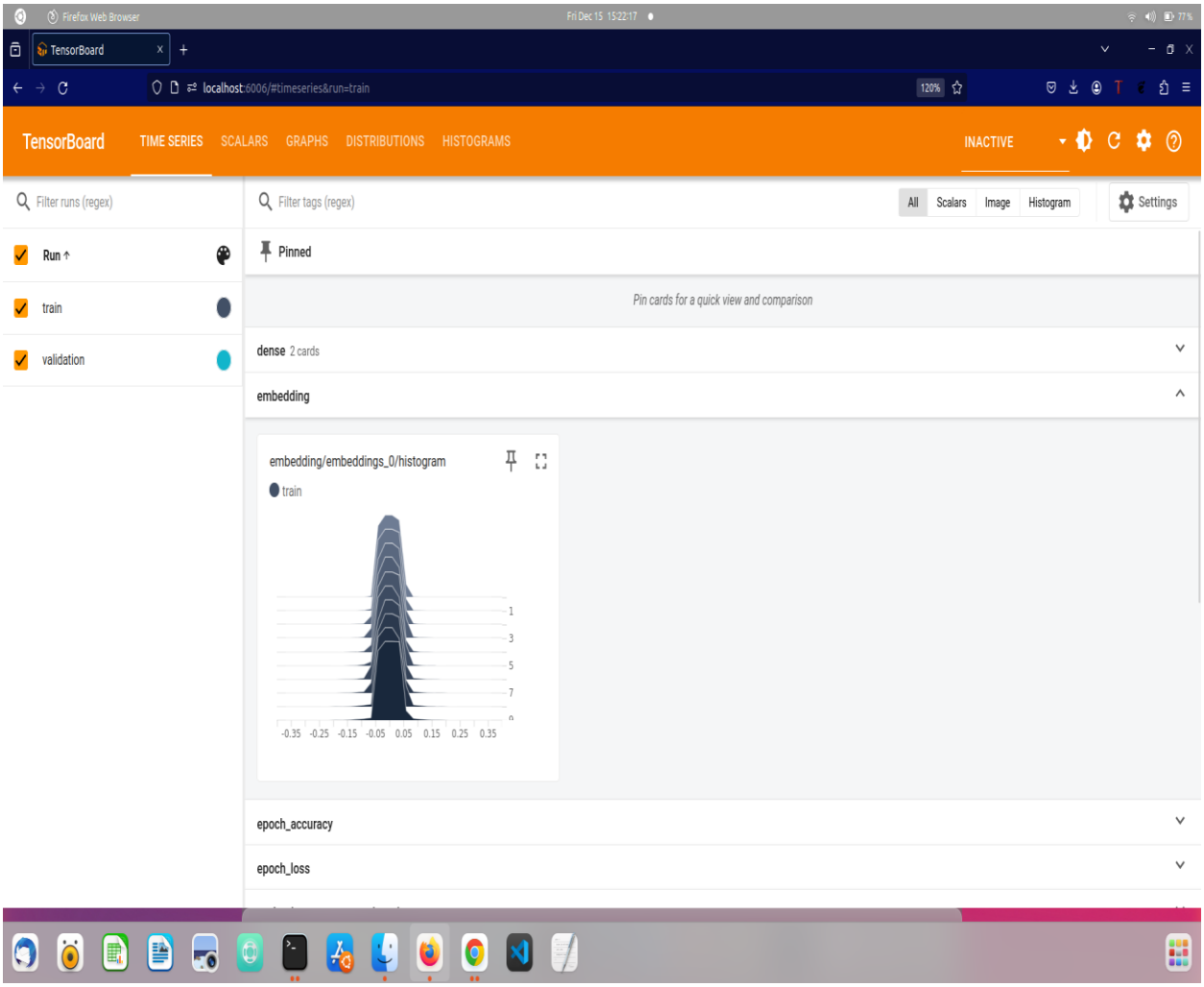

Firefox Web Browser
Fri Dec 15 15:23:14
TensorBoard
localhost:6006/#distributions&run=train
120%
TensorBoard
TIME SERIES
SCALARS
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Horizontal axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
train
validation
TOGGLE ALL RUNS
recipe_model
Filter tags (regular expressions supported)
dense
2
dense/bias_0/histogram
tag: dense/bias_0/histogram
train
0.03
0.01
-0.01
-0.03
-0.05
0 1 2 3 4 5 6 7 8 9
dense/kernel_0/histogram
tag: dense/kernel_0/histogram
train
0.1
0
-0.1
-0.2
-0.3
0 1 2 3 4 5 6 7 8 9
embedding
embedding/embeddings_0/histogram
tag: embedding/embeddings_0/histogram
train
0.3
0.1
-0.1
-0.3
0 1 2 3 4 5 6 7 8 9

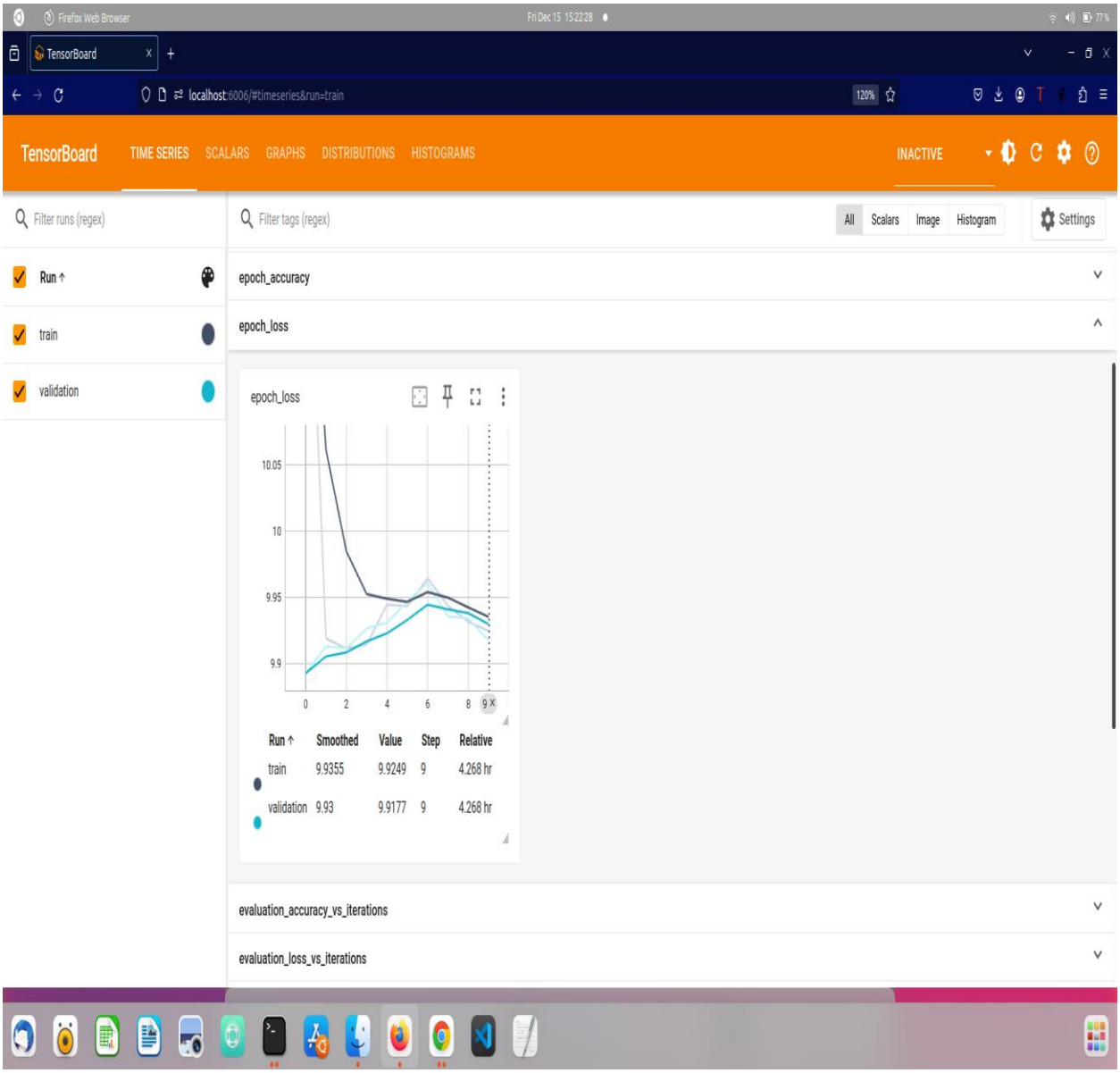

Firefox Web Browser
Fri Dec 15 15:23:16
TensorBoard
localhost:6006/#distributions&run=train
120%
TensorBoard
TIME SERIES
SCALARS
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Horizontal axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
train
validation
TOGGLE ALL RUNS
recipe_model
embedding
embedding/embeddings_0/histogram
tag: embedding/embeddings_0/histogram
train
0.3
0.1
-0.1
-0.3
0 1 2 3 4 5 6 7 8 9
lstm
3
lstm/lstm_cell/bias_0/histogram
tag: lstm/lstm_cell/bias_0/histogram
train
0.8
0.5
0.2
-0.1
lstm/lstm_cell/kernel_0/histogram
tag: lstm/lstm_cell/kernel_0/histogram
train
0.2
0.05
-0.1
-0.25
lstm/lstm_cell/recurrent_kernel_0/histogram
tag: lstm/lstm_cell/recurrent_kernel_0/histogram
train
0.35
0.2
0.05
-0.1
-0.25


Firefox Web Browser
Fri Dec 15 15:23:24
TensorBoard
localhost:6006/#histograms&run=train
120%
TensorBoard
TIME SERIES
SCALARS
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Histogram mode
OVERLAY
OFFSET
Offset time axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
train
validation
TOGGLE ALL RUNS
recipe_model
Filter tags (regular expressions supported)
dense
2
dense/bias_0/histogram
tag: dense/bias_0/histogram
train
-0.05 -0.03 -0.01 0.01 0.03
dense/kernel_0/histogram
tag: dense/kernel_0/histogram
train
-0.30 -0.20 -0.10 0.00 0.10 0.20
embedding
embedding/embeddings_0/histogram
tag: embedding/embeddings_0/histogram
train
-0.35 -0.25 -0.15 -0.05 0.05 0.15 0.25 0.35


Firefox Web Browser
Fri Dec 15 15:22:36
TensorBoard
localhost:6006/#timeseries&run=train
120%
TensorBoard
TIME SERIES
SCALARS
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Filter runs (regex)
Filter tags (regex)
All
Scalars
Image
Histogram
Settings
Run
train
validation
evaluation_loss_vs_iterations
evaluation_loss_vs_iterations
9.94
9.93
9.92
9.91
200
400
600
800
1,000
1,200
1,400
1,720
Run
Smoothed
Value
Step
Relative
validation
9.93
9.9177
1,720
4.275 hr
lstm 3 cards
lstm/lstm_cell/bias_0/histogram
train
lstm/lstm_cell/kernel_0/histogram
train
lstm/lstm_cell/recurrent_kernel_0/histogram
train


Firefox Web Browser
Fri Dec 15 15:23:26
TensorBoard
localhost:6006/#histograms&run=train
120%
TensorBoard
TIME SERIES
SCALARS
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Histogram mode
OVERLAY
OFFSET
Offset time axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
train
validation
TOGGLE ALL RUNS
recipe_model
embedding
embedding/embeddings_0/histogram
tag: embedding/embeddings_0/histogram
train
-0.35 -0.25 -0.15 -0.05 0.05 0.15 0.25 0.35
lstm
3
lstm/lstm_cell/bias_0/histogram
tag: lstm/lstm_cell/bias_0/histogram
train
-0.1 0.1 0.3 0.5 0.7 0.9 1.1
lstm/lstm_cell/kernel_0/histogram
tag: lstm/lstm_cell/kernel_0/histogram
train
-0.25 -0.15 -0.05 0.05 0.15 0.25
lstm/lstm_cell/recurrent_kernel_0/histogram
tag: lstm/lstm_cell/recurrent_kernel_0/histogram
train
-0.25 -0.15 -0.05 0.05 0.15 0.25 0.35


Firefox Web Browser
Fri Dec 15 15:22:17
TensorBoard
localhost:6006/#timeseries&run=train
120%
TensorBoard
TIME SERIES
SCALARS
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Filter runs (regex)
Filter tags (regex)
All
Scalars
Image
Histogram
Settings
Run
train
validation
Pinned
Pin cards for a quick view and comparison
dense 2 cards
embedding
embedding/embeddings_0/histogram
train
1
3
5
7
-0.35
-0.25
-0.15
-0.05
0.05
0.15
0.25
0.35
epoch_accuracy
epoch_loss



Conclusion:

We concluded that we were able to create a Food meal Recommendation System that is a user-friendly application that enhances the cooking experience and assists users in effectively preparing excellent meals by solving the problem that existed.