Analyzing Temporal, Spatial, and Historical Data in Rating Prediction Algorithms: A Comparative Study

***Abstract*— User reviews of a business have become an integral part of our quotidian lives. We decide places to visit, restaurants to dine, and services to utilize based on these reviews. Furthermore, the general population and their sentiments about a business have a huge impact on the performance of the business. In this project, we utilize the *Google Local Reviews* data set to predict the potential rating a user will give to a business. Specifically, we compare and contrast the effects of including temporal and spatial aspects of the data in our models for the task at hand.**

***Index Terms*— Google Business Reviews, Regression, Machine Learning, Deep Learning, Data Modelling, Information Systems, Personalized Recommendation.**

I. Introduction

The Google Local Reviews data set [1]–[3] consists of three files that contain information specific to users, businesses, and reviews, respectively, along with their metadata. Here, we intend to use this data set to predict what rating a person will give to a business based on *temporal*, *historical*, and *spatial* aspects of the data set.

We start by performing exploratory data analysis and data visualization to find any interesting trends, patterns, and seasonalities in the data. Based on that, we transform and prune our data. Further, we perform feature engineering on the pruned data set to create meaningful features for optimizing the regression problem at hand. We propose the design of several baseline (intuitive) models and a few machine and deep learning models, along with the evaluation metrics. Our aim is to create a model that fits the data and generates precise predictions.

Our report is structured as follows: Section 2 discusses existing literature on the topic; Section 3 details data pre-processing, exploratory data analysis, data visualization, and feature engineering techniques; Sections 4 and 5 define the predictive task and the design of models and experiments; and Section 6 concludes our report with results, outcomes, and future scope.

II. Literature Review

A. Data sets

In this report, we look specifically at the Google Local Reviews data set [1]–[4]. However, several similar data sets are available. These include:

1) Google Play Store Apps Reviews data set [5] which has been used in finding sentimental analysis of a particular app based on its reviews.
2) OpinRank Review data set [6] which contains users rating on cars and hotels around the world.

These data sets are crucial for businesses as they provide useful user trends which help businesses identify important user behaviours and trends.

B. Spatio-Temporal Models

Multiple studies like [20], [21] examine the effects of spatio-temporal data modelling for predictive tasks in detail. These studies show how effective such data can be in aiding the development of regression models ranging from prediction of housing prices to something more nuanced such as evolution of the urban system based on multiple socio-economic and geo-political factors. The work presented in these papers depict the statistical significance of both spatial and temporal features, and their results reflect a positive correlation between such features and the prediction.

Spatio-temporal data has also proven track record of being effective in reccomendation tasks. [22] proposes a Spatio-temporal approach to collaborative filtering. They propose Spatio-Temporal filtering (called ST-KF) which is an interesting variation of Kalman-Filtering. The promising results of the ST-KF approach in comparison to the User-Item-Covariance baseline, lead to the conclusion that Spatio-temporal data can be effectively harnessed for building significantly high-performing recommender systems as well.

Since we've established from literature that our spatio-temporal data preparation methodology is consistent with multiple lines of research, the problem of selecting an optimal regression model for the task still remains.

Authors in [23] examine the effects of metadata of 'user review text' on the 'users' rating of an item. They discuss at length about sentiment-analysis based on review content-structure and metadata and also provide interesting statistical insights on how regression models can effectively beat baseline models given some information about the review text structure. While the authors go on to discuss sentiment analysis of review text, we limit our scope to considering only certain aspects of the review text in our models.

On similar lines, [24] discuss about latent factors in text data, but more importantly, they extract the 'spatial' information from the text data to effectively train regression models. They draw correlations between the 'category', 'location', 'price range' of the place/business under review with the average rating given to a business. In our report, we discuss in detail how we attempt to exploit such correlations.

C. State-of-the-art Models

State-of-the-art models found for understanding spatial and temporal data sets include Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) models. These models can learn highly complex correlations between features very well and adapt themselves based on changes in locations and timestamps. This is something which simple machine learning models fail to do. In this report, we look at Deep Neural Network models, however, RNNs and LSTMs are out of scope.

III. Exploratory Data Analysis & Insights

In this section, we describe the exploratory data analysis carried out on the raw version of the Google Local Reviews data set (hereinafter referred to as 'data set'). Here, our aim is to bring about the notions of data completeness, sparseness, and cardinality to optimize our next steps. Additionally, we intend to find any interesting trends, patterns, and seasonalities in the data.

A. Data Pre-processing

Following pre-processing operations were performed on the dataset:

1) We merge the 'user', 'places' and 'review' data tables from the raw data set, using the foreign keys: `gPlusUserId` and `gPlusPlaceId`, and obtain a 'merged' table. As shown in Figure 1, it has 11 columns: `userName`, `gPlusUserId`, `gPlusPlaceId`, `rating`, `reviewText`, `categories`, `unixReviewTime`, `name`, `price`, `gps`, and `closed`.

Columns	Number of Nulls
`userName`	0
`gPlusUserId`	0
`rating`	0
`reviewText`	2649295
`categories`	135813
`gPlusPlaceId`	0
`unixReviewTime`	42698
`name`	0
`price`	6782945
`gps`	27431
`closed`	0

TABLE I
DISTRIBUTION OF NULLS

2) We define 3 new terms:
   - **Completeness**: Fraction of non-null values over total values.
   - **Distinctness**: Fraction of distinct non-null values over total non-null values.
   - **Uniqueness**: Fraction of unique values over the number of all non-null values of a column. Unique values occur exactly once. For example, $[a, a, b]$ contains one unique value b, so uniqueness is $1/3$.

   We drop all the columns that have extremely low completeness values or extremely high uniqueness values as they would skew our data and could result in sub-optimal models. We further prune the data set by dropping columns with excessive `NaN` values and perform `NaN` imputation on relevant columns like `gps`. The resulting dataframe has $8,649,011$ rows and 11 columns.
3) We expand the `gps` column to `lat` and `lon` columns indicating the latitude and the longitude of the place. We filter these latitudes and longitudes to keep only the businesses present in the United States of America. Now, we have a cleaned data set, called the 'pruned' data set, with $1,299,688$ rows and 13 columns.

We utilize this pruned data set and geo-locations to create a mapping of the various types of ratings that we see across all businesses/places in the US.

B. EDA and Outlier Detection

1) We compute the completeness, distinctness, and uniqueness statistics (as mentioned in previous section) on the entire data set and on every column in the data set. We also compute the following basic stats (Tables I, II, III, and IV):

	userName	gPlusUserId	rating	reviewText	categories	gPlusPlaceId	unixReviewTime	name	price	gps	closed
0	an lam	100000010817154263736	3.0	Chất lượng tạm ổn	[Giải Trí - Café]	108103314380004200232	1.372687e+09	Cà Phê Thăng Long	None	[10.852044, 106.65971]	False
1	hoang long nguyen	101659842775092396018	5.0	Good coffee, nice and peaceful place	[Giải Trí - Café]	108103314380004200232	1.354888e+09	Cà Phê Thăng Long	None	[10.852044, 106.65971]	False
2	Hong Le	107574994242995460712	2.0	Cho heo	[Giải Trí - Café]	108103314380004200232	1.352015e+09	Cà Phê Thăng Long	None	[10.852044, 106.65971]	False
3	HALİL TURGUT	100000013500285534661	5.0	Wc si temiz duzenli..	[Turkish Cuisine]	102194128241608748649	1.342871e+09	Selale Restaurant	None	[37.8037, 29.2209]	False
4	Akudosoft Yazılım	105271324704942360981	5.0	None	[Turkish Cuisine]	102194128241608748649	1.373148e+09	Selale Restaurant	None	[37.8037, 29.2209]	False

Fig. 1. Merged Dataframe

	`rating`	`unixReviewTime`	`reviewTextLength`
count	1299688	1299688	1299688
mean	3.84271	1353058634.13	237.3618
std	1.2784	39883090.8361	307.9316
min	0	662601600.0	0
25%	3.0	13394857034	60.0
50%	4.0	13620848764	139.0
75%	5.0	13774872745	294.0
max	5.0	13960723151	4090.0

TABLE II

BASIC STATS DISTRIBUTION

Completeness	57.04435
Missingness	42.9556

TABLE III

COMPLETENESS STATS

Column	**Entropy**
`rating`	1.4216
`unixReviewTime`	13.9288
`reviewTextLength`	6.4236

TABLE IV

DISTRIBUTION OF ENTROPY

a) Mean
b) Standard deviation
c) Minimum
d) Maximum
e) Percentiles
f) Entropy

We infer that columns with high entropy (high degree of randomness) can be removed. Moreover, we can perform data normalization based on the min. and max. values of each column.

2) Next, we perform outlier detection to remove skewed entries in our data. We employed the following outlier detection techniques:

a) **Z-score outliers**: Here, we compute the z-score on the `reviewTextLength` feature (details in subsequent sections). z-score is defined as: $z = (x - \mu)/\sigma$, where x is the data point, μ is its mean value, and σ is the standard deviation. We retain only the rows where $-2 \leq$ z-score ≤ 2. For example, users who give only 1-10 character reviews (very short) or 1000-2000 (very long) character reviews are usually junk reviews and can bias our model extensively. Such reviews are considered outliers and are removed.

b) **99-100 percentile outliers**: The data contains 99-100 percentile outliers if the difference between the 99.999^{th} percentile and 100^{th} percentile exceeds a certain threshold. We remove all the data points that have `reviewTextLength` exceeding this threshold.

c) **Cardinality based pruning**: We drop columns with extremely low ($\sim$ 0) and extremely high cardinalities ($\sim$ 1 million) based on empirically-chosen thresholds.

d) **Timestamp based filtering**: We noticed that there were some extremely old and irrelevant/out-of-trend reviews, such as a review from 1990 and a review from 2001, while the most recent review for the specific place was circa 2014. It makes sense to keep only the most recent reviews and not reviews that were given $\sim$ 2 decades ago.

C. Data Visualizations & Inferences

In this step, we utilize data visualization techniques to derive meaningful patterns in our data. The resulting inferences are subsequently used in data set localization and feature extraction/engineering.

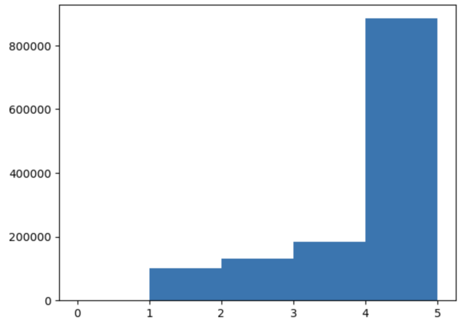

Fig. 2. Rating distribution cross-population

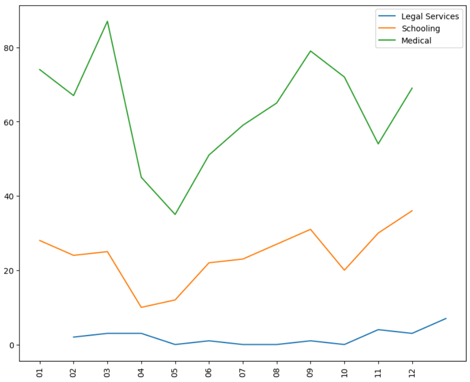

Fig. 3. Monthly Trends in Rating of Various Services

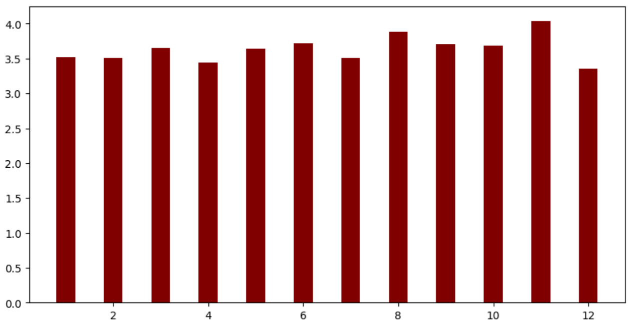

Fig. 4. Month Wise Rating Distribution in U.S.

1) Majority of the `ratings` ($> 62.5\%$) tend to be on the higher side of 4-5 stars (Figure 2). This indicates that users tend to be more biased towards higher ratings than lower.
2) We see from the world map distribution (Figure 11) that there is a dense set ($\sim$ 8 million) of ratings in the United States alone, which we can further localize into states to obtain state-wise business/place ratings.

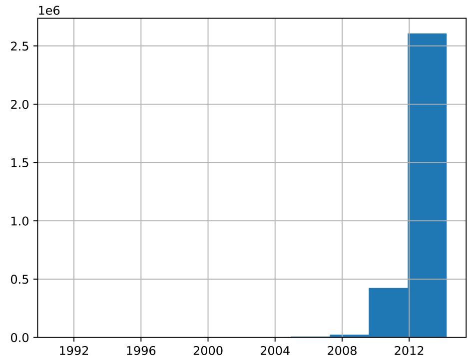

Fig. 5. No. of Ratings available year-wise

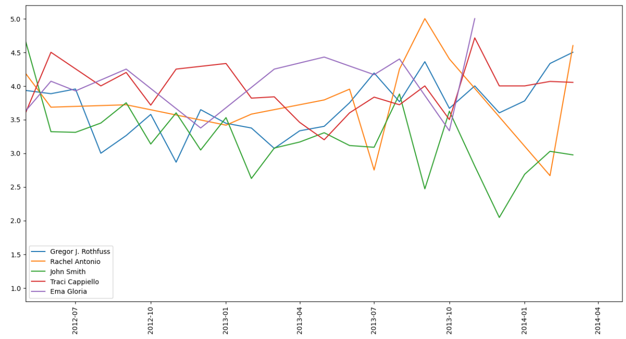

Fig. 6. User rating trends for a subset of users

3) The `categories` column has a strong correlation on the number of people that rate a business and the influence of categories is dynamic over time (Figure 3).
4) We see from temporal visualizations that the ratings tend to vary drastically based on the time of the year (Figures 4 & 5). This means that the `year-month` feature is a strong indicator of user ratings.
5) The `categories` can be clubbed together. By training a *word2vec* ([7], [8]) model on the temporally sorted business categories, we obtain an N-dimensional vector for each category in our data set. Further, running clustering on top of it, we group several categories into a single category.
6) The location of a specific business has a huge impact on the business' rating (Figures 12 & 13), on the business' performance (i.e., the chances of a business getting shut down), and on user sentiments.
7) It is also evident from the visualizations that the sentiments/ratings of 2 individuals for the same place/business in a large populous may vary

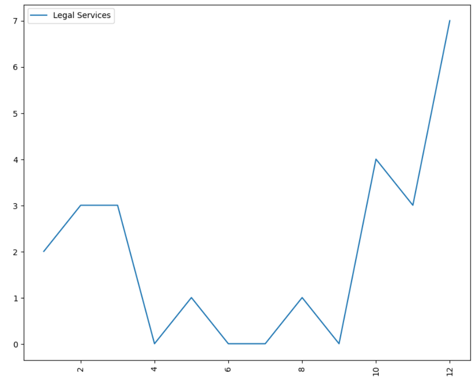

Fig. 7. Monthly Trends in Rating of Legal Services

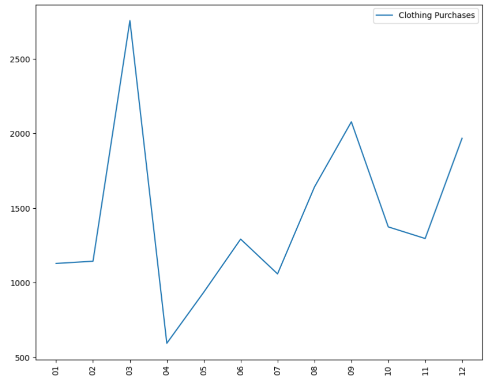

Fig. 8. Monthly Trends in Rating of Cloth Purchases

drastically based on their tastes (Figure 6).

8) Demand, utilization and rating of certain services or businesses such as ‘Gift Shops’, ‘Hospitals’, ‘Schools’, and ‘Legal Services’ vary greatly based on the time of the year. For instance, people tend to utilize legal services during the end of the financial year (Figure 7), whereas the cloth purchases are higher during the festive seasons (Figure 8).

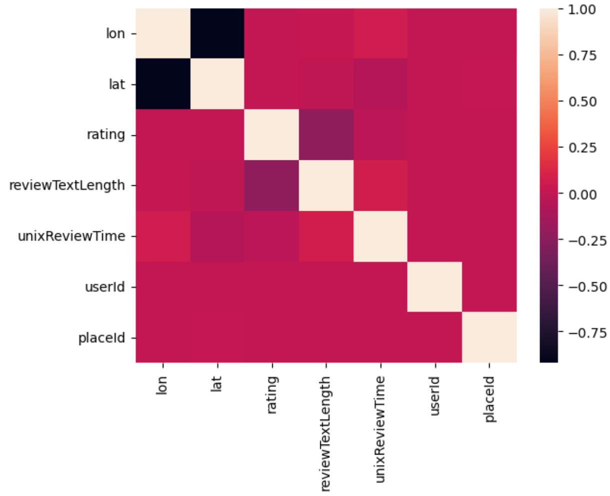

Fig. 9. Correlation Map of features for all data

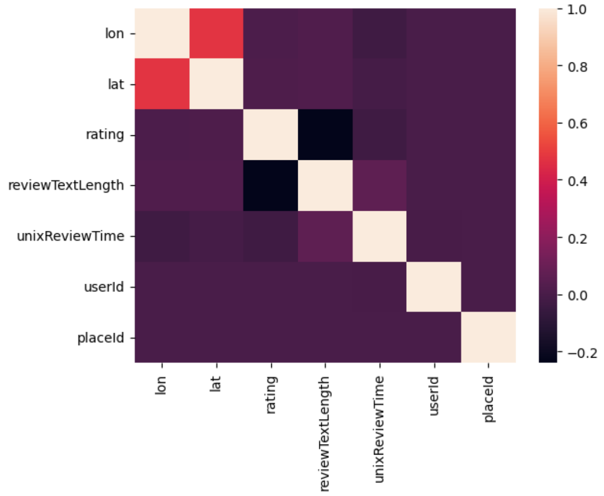

Fig. 10. Correlation Map of features of ratings data across the US

D. Feature Engineering

1) First, we reduce the scope of our data set to account for only interactions (i.e., user-business ratings) that occurred within the United States. This gave us about ~ 3 million data points to work with. We achieve this by constraining the latitudes and longitudes of the GPS data.

2) We write python scripts to find the *Geospatial distance* in miles from the latitude and longitude of our business/place to the center (latitude and longitude) of each state in the US. Using the nearest neighbour approach, we zero-in on the closest state in the US for the said business. Thus, we add new column `state` as an indicator of the rating given to a place.

3) Since it is evident from our temporal correlation analysis that the column `unixReviewTime` is highly relevant, we extract the `year-month` information from it (in `YYYY-MM` format) and create `month` and `year` features.

4) In our EDA on the `categories` column, we had shown that several categories can be clubbed into a single category. For every cluster identified, we resort to a customized category grouping technique using ‘word’ similarities (i.e., of all the categories that a particular place/business belongs to, which category best describes the place). We posit that there exists 10 such mutually

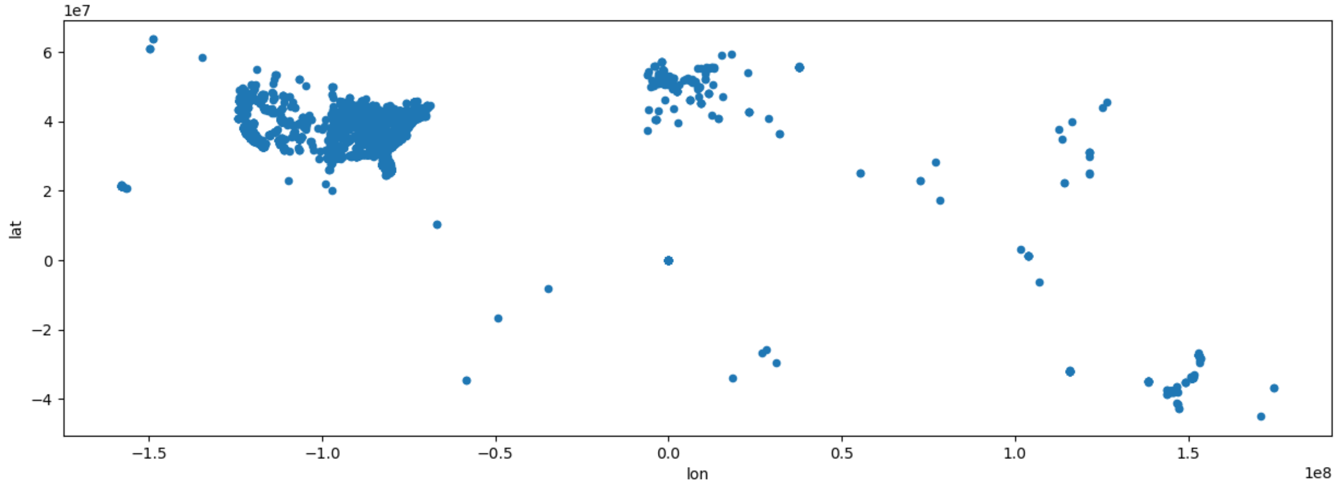

Fig. 11. World Map based on GPS locations of Businesses

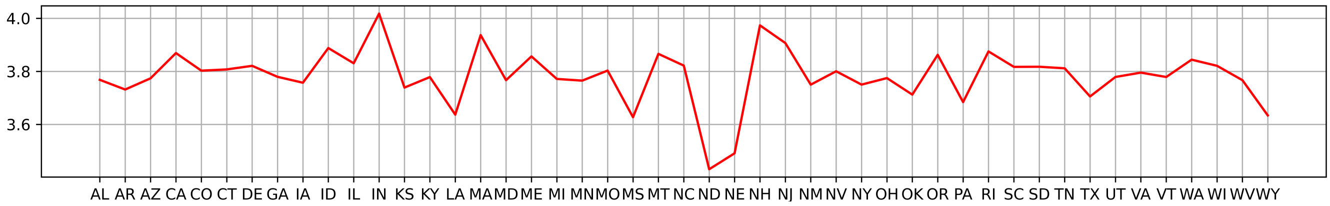

Fig. 12. US State-wise Average Rating

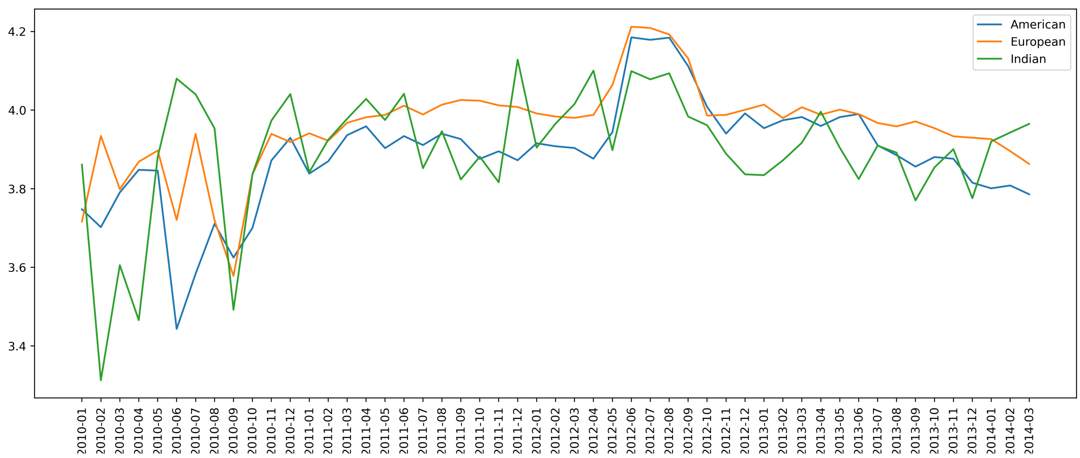

Fig. 13. Temporal Rating Trends in 3 Types of Restaurants

exclusive categories that we can optimally ‘bin’ our places into. These are as follows: ‘Associations/Organizations’, ‘Entertainment’, ‘Legal Services’, ‘Medical’, ‘Public’, ‘Restaurant’, ‘School’, ‘Shops and Stores’, ‘Venues’, and ‘Others’.

5) Additionally, our category-wise user rating prediction analysis in the visualizations showed a strong dependency of average user rating for a category with the places that belong to those categories. Therefore, we systematically generated the ‘User Average Ratings’ for every business grouped ‘Category-Wise’ and included that as another data point in the parameter vector.
6) Finally, we perform a one-hot-encoding

of the columns `year`, `month`, `state`, `final-category`.

The resulting data set has 84 columns: `gPlusUserId`, `gPlusPlaceId`, `userCategoryAvgRating`, `month_[01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12]`, `year_[1990, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014]`, `final_category_[AssocOrgs, Entertainment, Legal, Medical, Others, Public, Restaurant, School, ShopsStores, Venues]`, and `state_[AL, AR, AZ, CA, CO, CT, DE, GA, IA, ID, IL, IN, KS, KY, LA, MA, MD, ME, MI, MN, MO, MS, MT, NC, ND, NE, NH, NJ, NM, NV, NY, OH, OK, OR, PA, RI, SC, SD, TN, TX, UT, VA, VT, WA, WI, WV, WY]`.

IV. Predictive Modelling & Tasks Identification

Following are a few possible predictive tasks for our data set:

1) Given user and place metadata, predict what rating a user would give to a business.
2) Given the location and historical user ratings of a place, predict whether a business would fail or succeed.
3) Predicting the top-*k* most similar users for a particular user given the way they rated different places in different categories.

In this report, we discuss the implementation, experiments and analysis of the first task.

Problem Statement: Predict what rating a user will give to a business based on the time of year (the *temporal* aspect of the data set), the past ratings of the user (the *historical* aspect of the data set), and the geographical coordinates of the business (the *spatial* aspect of the data set).

We propose regression-based techniques to develop solutions to this problem as we need to make real-valued predictions in the range 1-5 (star ratings). Multivariate regression is the ideal way to approach this problem. So, initially, we start by building some intuitive baselines and then suggest a few machine and deep learning models. We perform a comparative analysis of the performance of these models by evaluating them on suitable evaluation metrics. We also attempt to avoid over-fitting by exploring various techniques of regularization. Finally, we explore cold-start issue as well.

V. Model Selection & Experiments

A. Experimental Setup

1) Data is split into 3 subsets: 'train', 'validation', and 'test' in the ratio $60 : 20 : 20$.
2) Regularization is done wherever applicable ensuring to optimize the validation loss.
3) If the loss between any two subsequent epochs is less than 0.0001 (tolerance value), we argue that the model has converged.
4) Simpler models are tried first and then, more intricate models.

B. Baseline Models

1) Always Predict Mean: Our first naive baseline is predicting the mean rating of all businesses for all users satisfying a particular condition, such as falling within a specific state or category. The different variants of this baseline are as follows:

- **Always Predict Mean Per State**: Find average rating of all businesses per `state` and return this value as the predicted rating based on the `state` of the input test data point.
- **Always Predict Mean Per Category**: Do the same as above but based on `category` of the test data point.
- **Always Predict Mean Per Unit of Time**: Do the same as above but based on `month` and `year` of the test data point.

2) Based on Most Popular: An alternative baseline can be to always predict the mean rating of the most popular place/business satisfying a particular condition. Here, we hypothesize that a place/business is popular if it has the most number of reviews. The different variants of this baseline are as follows:

- **Based on Most Popular Per State**: Find the average rating for the most popular place in every `state` and return this value as the predicted rating based on the `state` of the input test data point.
- **Based on Most Popular Per Category**: Do the same as above but based on `category` of the test data point.
- **Based on Most Popular Per Unit of Time**: Do the same as above but based on `month` and `year` of the test data point.

Table V summarizes the baseline performances.

Baseline Model	Type of Data Passed to Model	MSE	MAE
Always Predict Mean			
Per state	No Spatial or Temporal Features	2.5374	1.3143
	SpatioTemporal Features (No Timestamp sorting)	1.6746	1.6709
	Temporally Sorted Spatial	2.5295	1.312
Per category	No Spatial or Temporal Features	2.5374	1.3143
	SpatioTemporal Features (No Timestamp sorting)	1.673	1.067
	Temporally Sorted Spatial	2.518	1.312
Per unit of time	No Spatial or Temporal Features	2.5374	1.3143
	SpatioTemporal Features (No Timestamp sorting)	1.6616	1.0711
	Temporally Sorted Spatial	N/A	N/A
Based on Most Popular			
Per State	No Spatial or Temporal Features	2.5374	1.3143
	SpatioTemporal Features (No Timestamp sorting)	2.3346	1.1808
	Temporally Sorted Spatial	2.8617	1.3497
Per category	No Spatial or Temporal Features	2.5374	1.3143
	SpatioTemporal Features (No Timestamp sorting)	1.8017	1.0675
	Temporally Sorted Spatial	2.7617	1.3474
Per unit of time	No Spatial or Temporal Features	2.5374	1.3143
	SpatioTemporal Features (No Timestamp sorting)	2.6878	1.198
	Temporally Sorted Spatial	N/A	N/A

TABLE V
BASELINE MODEL PERFORMANCE

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (484576, 512)             43520

 dropout (Dropout)           (484576, 512)             0

 dense_1 (Dense)             (484576, 256)             131328

 dropout_1 (Dropout)         (484576, 256)             0

 dense_2 (Dense)             (484576, 64)              16448

 dropout_2 (Dropout)         (484576, 64)              0

 dense_3 (Dense)             (484576, 16)              1040

 dropout_3 (Dropout)         (484576, 16)              0

 dense_4 (Dense)             (484576, 1)               17

=================================================================
Total params: 192,353
Trainable params: 192,353
Non-trainable params: 0
_________________________________________________________________
```

Fig. 14. TensorFlow DNN Model

C. Machine & Deep Learning Models

1) **Linear Regression**: Simple multi-variate linear regression with no regularization. We use `sklearn.linear_model.LinearRegression` here [16].
2) **Ridge Regression with Regularization**: A variant of linear regression which regularizes given the L2-norm. Here, we find the optimal `alpha` (regularization term) by tuning the same on our validation data set. This comes out to be 100. We use `sklearn.linear_model.Ridge` here [17].
3) **Random Forest Regression**: Random forest regression with the following hyper-parameters: `n_estimators`: 100 and `min_samples_split`: 250. We use `sklearn.ensemble.RandomForestRegressor` here [19].
4) **XGBoost Regression**: A Gradient Boosted Decision Tree (GBDT) algorithm that does extremely well for sparse data. We use `xgboost.XGBRegressor` here [18], with `reg_lambda`: 2.0 and `learning_rate`: 0.1.

Regression Model	Type of Data Passed to Model	MSE	MAE
Linear Regression	No Spatial or Temporal Features	Same as Baseline	Same as Baseline
	SpatioTemporal Features (No Timestamp sorting)	0.4172	0.3567
	Temporally Sorted Spatial	**0.3610**	**0.2772**
Ridge Regression with Regularization	No Spatial or Temporal Features	Same as Baseline	Same as Baseline
	SpatioTemporal Features (No Timestamp sorting)	0.4172	0.3565
	Temporally Sorted Spatial	**0.3610**	**0.2772**
Random Forest	No Spatial or Temporal Features	Same as Baseline	Same as Baseline
	SpatioTemporal Features (No Timestamp sorting)	0.4245	0.3616
	Temporally Sorted Spatial	0.3659	0.2751
XGBoost	No Spatial or Temporal Features	Same as Baseline	Same as Baseline
	SpatioTemporal Features (No Timestamp sorting)	0.4173	0.3555
	Temporally Sorted Spatial	0.3625	0.2738
Bayesian Personalized Ranking	No Spatial or Temporal Features	Same as Baseline	Same as Baseline
	SpatioTemporal Features (No Timestamp sorting)	1.5905	1.02350
	Temporally Sorted Spatial	2.5542	1.3101
BPR with Latent Factors	No Spatial or Temporal Features	Same as Baseline	Same as Baseline
	SpatioTemporal Features (No Timestamp sorting)	1.6740	1.06955
	Temporally Sorted Spatial	2.5354	1.3140
TensorFlow – Deep Neural Network	No Spatial or Temporal Features	Same as Baseline	Same as Baseline
	SpatioTemporal Features (No Timestamp sorting)	0.4166	0.3654
	Temporally Sorted Spatial	**0.3615**	**0.2737**

TABLE VI
PERFORMANCES OF VARIOUS REGRESSION MODELS

5) **Bayesian Personalized Ranking (BPR)**: A personalized ranking technique, where we pass only the user-place interaction pairs and the model learns user-features based on the rating that a user gave to a specific place. This is a simple model with just user/place biases (α, β_u, and β_i).
6) **BPR with Latent Factors**: This model not only takes in the user/place biases but also utilizes a γ term which models the latent factors of users and items. The resulting function we train has the parameters α, β_u, β_i, γ_u, and γ_i.
7) **TensorFlow DNN**: A feed-forward deep-neural network trained on TensorFlow with 5 dense layers and 4 dropout layers for regularization [9], [10]. The sequential model is described in Figure 14.

D. Evaluation Metrics

Since our task involves regression, we use the following evaluation metrics:

- **Mean Absolute Error (MAE)**: This is the arithmetic average of absolute errors, i.e., the absolute difference between the model's predicted rating and the actual rating, *viz.*, MAE $= \sum_{n=1}^{N} |\hat{y} - y|/N$
- **Mean Squared Error (MSE)**: This is the arithmetic average of the square of errors, i.e., the square of the difference between the model's predicted rating and the actual rating, *viz.*, MSE $= \sum_{n=1}^{N} (\hat{y} - y)^2/N$. MSE penalizes heavily for large deviations and very little for smaller deviations.

Table VI depicts the performance of each of these models on the test data set.

E. Cold Start Problem

In the data set, almost 70% of reviews are given by unique users, i.e., a large number of users have reviewed only 1 place/business. Predicting the rating for these users is a massive challenge as we do not have extensive information about them. Any similarity function will not work as there is no feature through which we can compare users. Therefore for a new user, we predict the mean rating of the business as the potential rating.

VI. Results & Conclusion

A. Results & Outcomes

From our experimental analyses, we infer that several regression algorithms perform reasonably well on the data set. We see that the *Tensorflow based Deep Neural Network with Dropout Regularization* performs the best on the data if MAE is considered as the evaluation metric. In terms of MSE, *XGBoost Regressor* is the best-performing model. While the performance of all models are reasonably close, we see that Bayesian Personalized Ranking both with and without Latent Factors perform worse on the data. This is predominantly due to the fact that our data is sparse (as detailed in the Cold Start section), which renders a personalized ranking strategy ineffective.

Furthermore, we see that the performance of every model improves when we provide temporal, spatial as well as historical data. This is in line with our reasoning that additional metadata improves the predictive power of the regression model.

B. Conclusion & Future Work

We explored several data cleaning, pruning, and encoding techniques. Our data visualizations and inferences allowed to extract essential features and create several baselines. We designed a few machine and deep learning regression models and tried overfitting by introducing regularization wherever necessary. We also handled the cold-start problem.

Future work on the realm of predictive analysis and modelling on the data set could include the predicting if places/businesses will shut down, predicting similar users, developing recommendation engines based on user ratings, modelling the review text data, and finding N-most-similar reviews/places based on user sentiment and N-most similar places based on the place description and category.

In conclusion, we believe that multiple inherent characteristics exist in the data set that can be put to great use in development of future models. We see that identifying and exploiting subtle relations and correlations in the data go a long way in the development of optimal machine learning-based predictive models.

References

[1] Pasricha, Rajiv, and Julian McAuley. "Translation-based factorization machines for sequential recommendation." Proceedings of the 12th ACM Conference on Recommender Systems. 2018.

[2] He, Ruining, Wang-Cheng Kang, and Julian McAuley. "Translation-based recommendation." Proceedings of the eleventh ACM conference on recommender systems. 2017.

[3] `https://cseweb.ucsd.edu/~jmcauley/datasets.html#google_local`

[4] `https://developers.google.com/search/docs/appearance/structured-data/local-business`

[5] `https://www.kaggle.com/datasets/prakharrathi25/google-play-store-reviews`

[6] `https://archive.ics.uci.edu/ml/datasets/opinrank+review+dataset`

[7] Mikolov, Tomas; et al. (2013). "Efficient Estimation of Word Representations in Vector Space". arXiv:1301.3781 [cs.CL].

[8] Mikolov, Tomas; Sutskever, Ilya; Chen, Kai; Corrado, Greg S.; Dean, Jeff (2013). Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems. arXiv:1310.4546. Bibcode:2013arXiv1310.4546M.

[9] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. arXiv:1606.07792 [cs.LG]

[10] `https://www.tensorflow.org/tutorials/keras/regression`

[11] Singh, Ruchi, Yashaswi Ananth, and Dr Jongwook Woo. "Big data analysis of local business and reviews." Proceedings of the International Conference on Electronic Commerce. 2017.

[12] He, Ruining, Wang-Cheng Kang, and Julian J. McAuley. "Translation-based Recommendation: A Scalable Method for Modeling Sequential Behavior." IJCAI. 2018.

[13] Muñoz, Jesús, and Ángel M. Felicísimo. "Comparison of statistical methods commonly used in predictive modelling." Journal of Vegetation Science 15.2 (2004): 285-292.

[14] `https://geopandas.org/en/stable/docs/reference.html`

[15] `https://seaborn.pydata.org/`

[16] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html`

[17] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html#sklearn.linear_model.Ridge`

[18] `https://xgboost.readthedocs.io/en/stable/python/python_api.html`

[19] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`

[20] Frazier, Christopher, and Kara M. Kockelman. "Spatial econometric models for panel data: incorporating spatial and temporal data." TRR 1902.1 (2005): 80-90.

[21] Chica-Olmo, Jorge, Rafael Cano-Guervos, and Mario Chica-Rivas. "Estimation of housing price variations using spatio-temporal data." Sustainability 11.6 (2019): 1551.

[22] Lu, Zhengdong, Deepak Agarwal, and Inderjit S. Dhillon. "A spatio-temporal approach to collaborative filtering." Proceedings of 3rd ACM conference on Recommender systems. 2009.

[23] Ganu, Gayatree, Noemie Elhadad, and Amélie Marian. "Beyond the stars: improving rating predictions using review text content." WebDB. Vol. 9. 2009.

[24] Wang, Hongning, Yue Lu, and Chengxiang Zhai. "Latent aspect rating analysis on review text data: a rating regression approach." Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. 2010.