Project: Conversation Dialogue Identification

York University – CSML1010 – Final Project – Group 3: Jerry Khidaroo, Paul Doucet

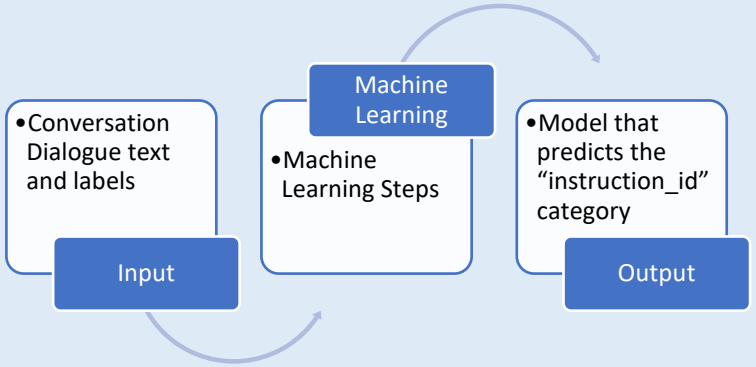
Instructor: Dr. Annie En-Shiun Lee

YORK UNIVERSITÉ UNIVERSITY

Final Project: Machine Learning Life-cycle
Introduction
Existing Work
Methodology
Results
Discussion
Conclusion
References/Bibliography

Project Definition

Existing Work



Project Definition: Conversation Identification

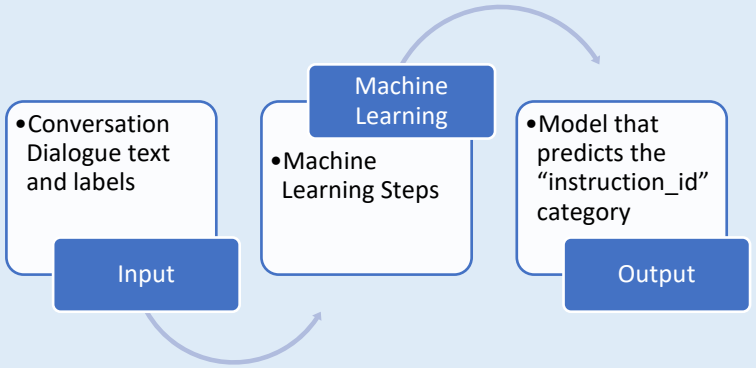
The problem we will examine is a supervised multi-class text classification problem.

Goal: *The goal is to investigate which supervised machine learning methods will give the best results in classifying the texts from our dataset into the pre-defined categories.*

- **Input:** conversation dialogue text and labels
- **Output:** model that predicts the "instruction_id" category correctly

Project Definition

Existing Work



Type	Conversation Dialogue Description
Auto-repair-appt-1	Users will pretend they need to take their car to the mechanic, so they need to get an appointment scheduled
Coffee-ordering-1	Users will pretend they've decided to order a coffee drink from a coffee shop
Coffee-ordering-2	Users will pretend they've decided to order a coffee drink, and makes changes to the drink after the initial options have been requested
movie-finder	User is looking for a movie to see at home
movie-tickets-1	User wants to see a movie playing now
movie-tickets-2	User wants to see a movie playing now, settling for a second choice
movie-tickets-3	User wants to see one of two movies
Pizza-Ordering-1	User orders one pizza, and ask all relevant details
Pizza-Ordering-2	User orders one pizza with two toppings, and ask all relevant details
restaurant-table-1	Users will pretend they are searching for a restaurant and book a table
restaurant-table-2	Users will pretend they are searching for a restaurant and book a table, and will need to find an alternative when choice is not available
restaurant-table-3	Users will pretend they are searching for a restaurant and book a table, and will look at options at two restaurants
uber-lyft-1	Users will pretend they need to order a car for a ride inside a city
uber-lyft-2	Users will pretend they need to order a car for a ride inside a city, and looking for an alternative when choice is not available

Project Definition

Existing Work

Existing Work	
1. Li, Susan, Feb 19, 2018, Multi-Class Text Classification with Scikit-Learn https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f	Gives some good examples of Data Exploration.
2. Bansal, Shivam, Jan. 12, 2017, Ultimate Guide to Understand and Implement Natural Language Processing (with codes in Python) https://www.analyticsvidhya.com/blog/2017/01/ultimate-guide-to-understand-implement-natural-language-processing-codes-in-python/	Has a great overview of many of the steps involved in NLP.
3. Gupta, Shikhar, Jun 12, 2018, Machine Learning Model Evaluation & Selection, Validation strategies for your machine learning model https://heartbeat.fritz.ai/model-evaluation-selection-i-30d803a44ee	Provides a summary of Model Selection and Validation strategies.
4. scikit-learn 0.23.0, 3.2. Tuning the hyper-parameters of an estimator https://scikit-learn.org/stable/modules/grid_search.html#grid-search	Explains what is involved in Hyperparamet Tuning
5. scikit-learn 0.22.2, Receiver Operating Characteristic (ROC) https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html	Provides examples of calculating and ploting ROC/AUC curves for Multi-Class Classification Models
6. Smolyakov, Vadim, Aug 22, 2017, Ensemble Learning to Improve Machine Learning Results, How ensemble methods work: bagging, boosting and stacking https://blog.statsbot.co/ensemble-learning-d1dcd548e936	Discusses 3 types of Ensemble models: Bagging, Boosting and Stacking.
7. Koehrsen, Will, May 18, 2018, A Complete Machine Learning Walk-Through in Python: Part Three, Interpreting a machine learning model and presenting results https://towardsdatascience.com/a-complete-machine-learning-walk-through-in-python-part-three-388834e8804b https://github.com/WillKoehrsen/machine-learning-project-walkthrough/blob/master/Machine%20Learning%20Project%20Part%203.ipynb	Gives a good explanation of Model Interpretability.

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Model Evaluation & Selection

Schema	Example Value
self-dialogs.json	
conversation_id	dlg-00055f4e-4a46-48bf-8d99-4e47766...
instruction_id	restaurant-table-2
utterances	
index	0
speaker	USER
text	Hi, I'm looking to book a table for Korean f...
segments	
end_index	49
start_index	13
text	Southern NYC, maybe the East Village
annotations	
name	restaurant_reservation.location.restauran...

Abc self-dialogs.json instruction_id	Abc self-dialogs.json text
pizza-ordering-2	Okay. Do you want any veggies added?
pizza-ordering-2	No thanks.
pizza-ordering-2	Okay, and I assume that you want the largest size since...
pizza-ordering-2	That's correct.
pizza-ordering-2	Alright, I've ordered you one large two topping pizza wi...
pizza-ordering-2	Perfect. thanks.
pizza-ordering-2	I have sent the receipt to your cell.
pizza-ordering-2	I see that. Thank you very much.
pizza-ordering-2	Your welcome, enjoy your dinner.
auto-repair-appt-1	I would like to schedule an appointment with Intelligen...
auto-repair-appt-1	Sure thing, what is the nature of your issue?
auto-repair-appt-1	I want my fuel system checked and inspected

The data consists of dialog conversations in JSON format. We plan to import the JSON data and parse the conversation texts into a dataframe that has the concatenated dialog texts and labels.
Next we will perform Data Cleaning and NLP to obtain a normalized corpus that has white-space characters and stop words removed.

Data Preparation

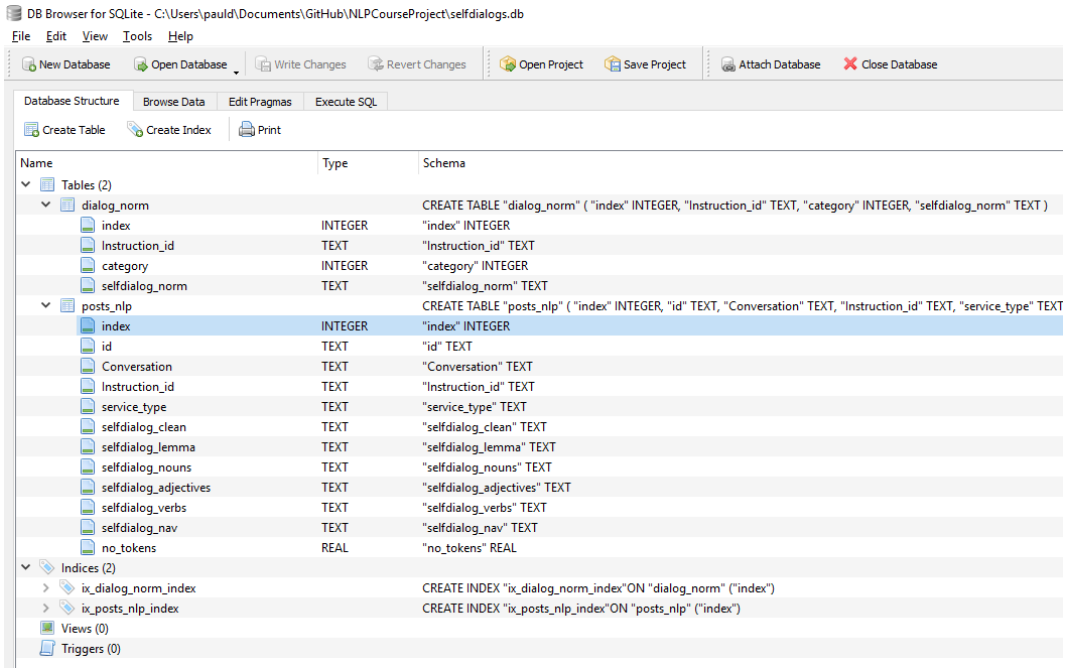
Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Model Evaluation & Selection

	id	Conversation	Instruction_id	service_type	selfdialog_clean	selfdialog_lemma	selfdialog_nouns	selfdialog_adjectives	selfdialog_verbs	selfdialog_nav	no_tokens
0	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	Hi, I'm looking to book a table for Korean fod...	restaurant-table-2	restaurant	Hi, I'm looking to book a table for Korean fod...	hi , -PRON- be look to book a table for korean...	table fod area southern nyc east village thurs...	korean what great great great fine available s...	be look book be think have be need do want sit...	table fod area southern nyc east village thurs...	192.0
1	dlg-0009352b-de51-474b-9f13-a2b0b2481546	Hi I would like to see if the Movie What Men W...	movie-tickets-1	movie	Hi I would like to see if the Movie What Men W...	hi -PRON- would like to see if the movie what ...	movie what men ticket friend ticket time today...	what that funny -PRON- much -PRON- own fine -P...	would like see want be play be show would like...	movie what men ticket friend ticket time today...	236.0
2	dlg-00123c7b-15a0-4f21-9002-a2509149ee2d	I want to watch avengers endgame where do you ...	movie-tickets-3	movie	I want to watch avengers endgame where do you ...	-PRON- want to watch avenger endgame where do ...	avenger endgame bangkok hotel time movie o'clo...	good what many other -PRON- new afraid interes...	want watch do want watch close stay sound do w...	avenger endgame bangkok hotel time movie o'clo...	150.0
3	dlg-0013673c-31c6-4565-8fac-810e173a5c53	I want to order a pizza from Bertuccis in Chel...	pizza-ordering-2	pizza	I want to order a pizza from Bertuccis in Chel...	-PRON- want to order a pizza from bertuccis in...	pizza bertuccis chelmsford ma what type pizza ...	what large different what large -PRON- large g...	want order would like understand will do have ...	pizza bertuccis chelmsford ma what type pizza ...	155.0
4	dlg-001d8bb1-6f25-4ecd-986a-b7eeb5fa4e19	Hi I'd like to order two large pizzas. Sure, w...	pizza-ordering-2	pizza	Hi I'd like to order two large pizzas. Sure, w...	hi -PRON- would like to order two large pizza ...	pizza kind pizza mind please anything meat lov...	large what hawaiian large sorry sure what -PRO...	would like order have will have be can get wou...	pizza kind pizza mind please anything meat lov...	257.0



Data Preparation

Cleaning and NLP

Data Exploration

- **Statistical Analysis**
- Exploring Word Clouds
- Exploring Complexity

Feature Engineering & Selection

Model Evaluation & Selection

	count	unique	top	freq
service_type	7708	6	pizza	1468
Instruction_id	7708	14	pizza-ordering-2	1211

	count	mean	std	min	25%	50%	75%	max
index	7708.00	3853.50	2225.25	0.00	1926.75	3853.50	5780.25	7707.00
no_tokens	7708.00	228.51	80.57	20.00	175.00	215.00	267.00	1336.00

Statistical Analysis

Data Preparation

Cleaning and NLP

Data Exploration

- Statistical Analysis
- **Exploring Word Clouds**
- Exploring Complexity

Feature Engineering & Selection

Model Evaluation & Selection

movie-finder

Restaurant-table-3

Exploring Word Clouds

- Created and displayed word clouds using the tokens created above and a second one

Data Preparation

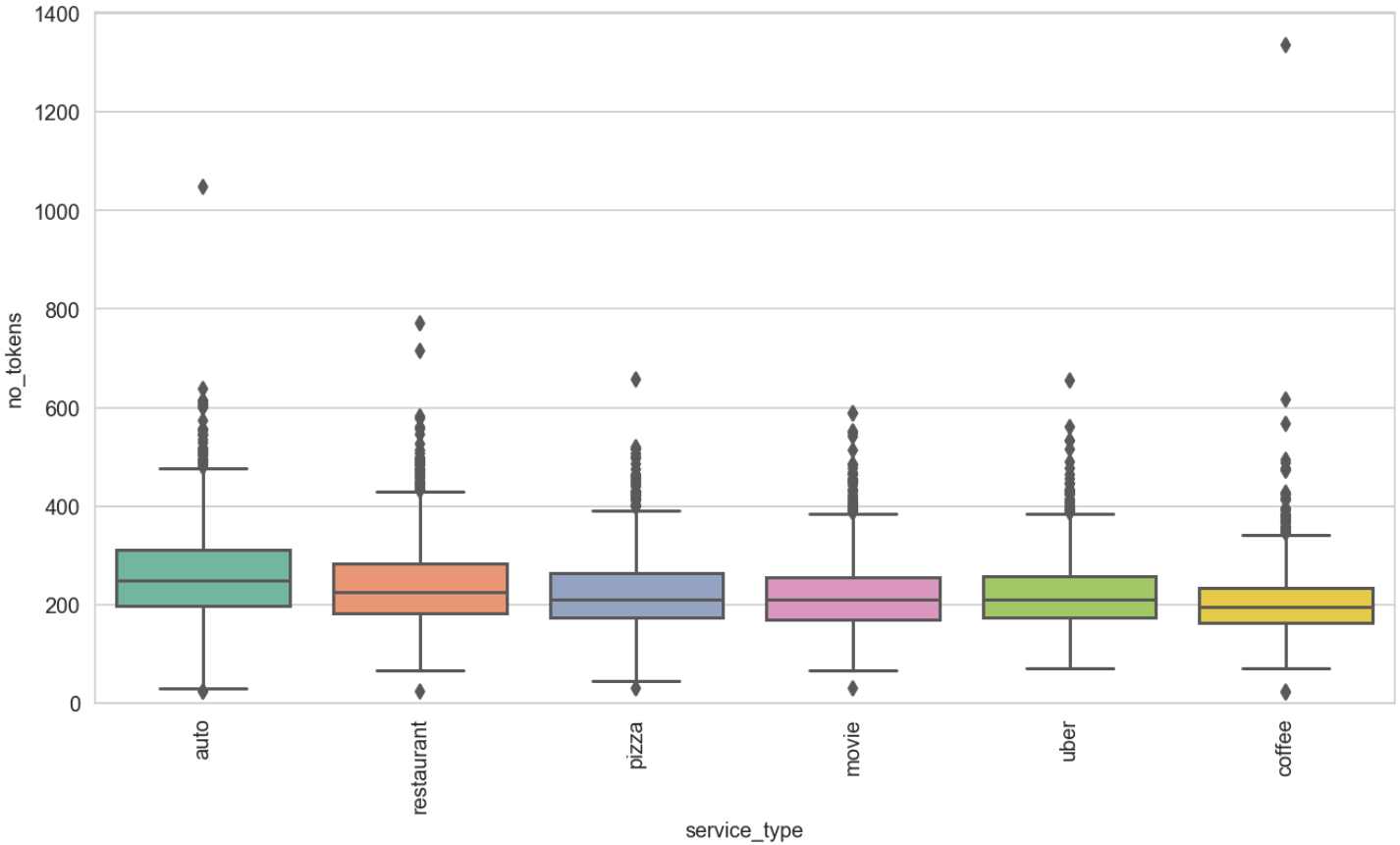
Cleaning and NLP

Data Exploration

- Statistical Analysis
- Exploring Word Clouds
- **Exploring Complexity**

Feature Engineering & Selection

Model Evaluation & Selection



Exploring Complexity

- Reviewed for service type: average number of tokens; distribution of tokens; printed outliers; compared # of tokens per ‘instruction_id’

Data Preparation

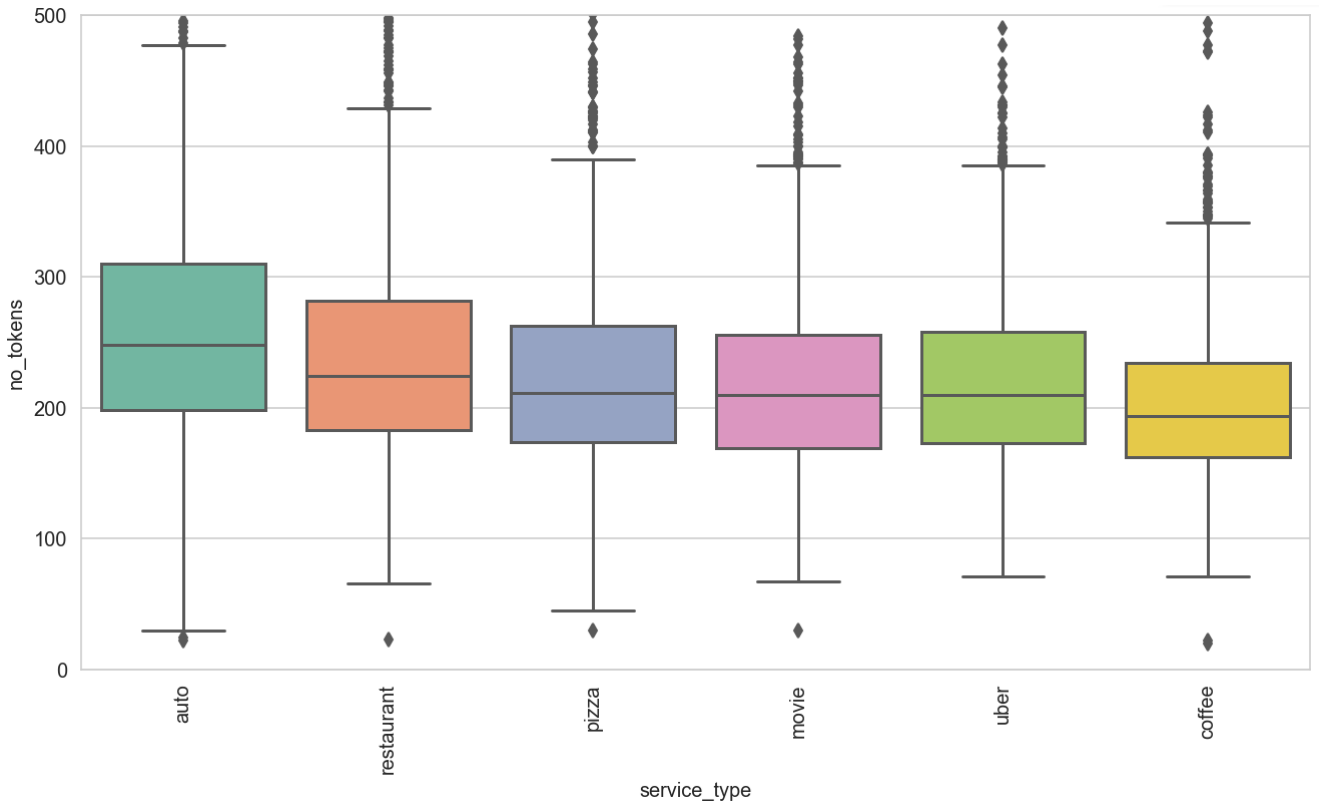
Cleaning and NLP

Data Exploration

- Statistical Analysis
- Exploring Word Clouds
- **Exploring Complexity**

Feature Engineering & Selection

Model Evaluation & Selection



Exploring Complexity

- Reviewed for service type: average number of tokens; distribution of tokens; printed outliers; compared # of tokens per 'instruction_id'

Data Preparation

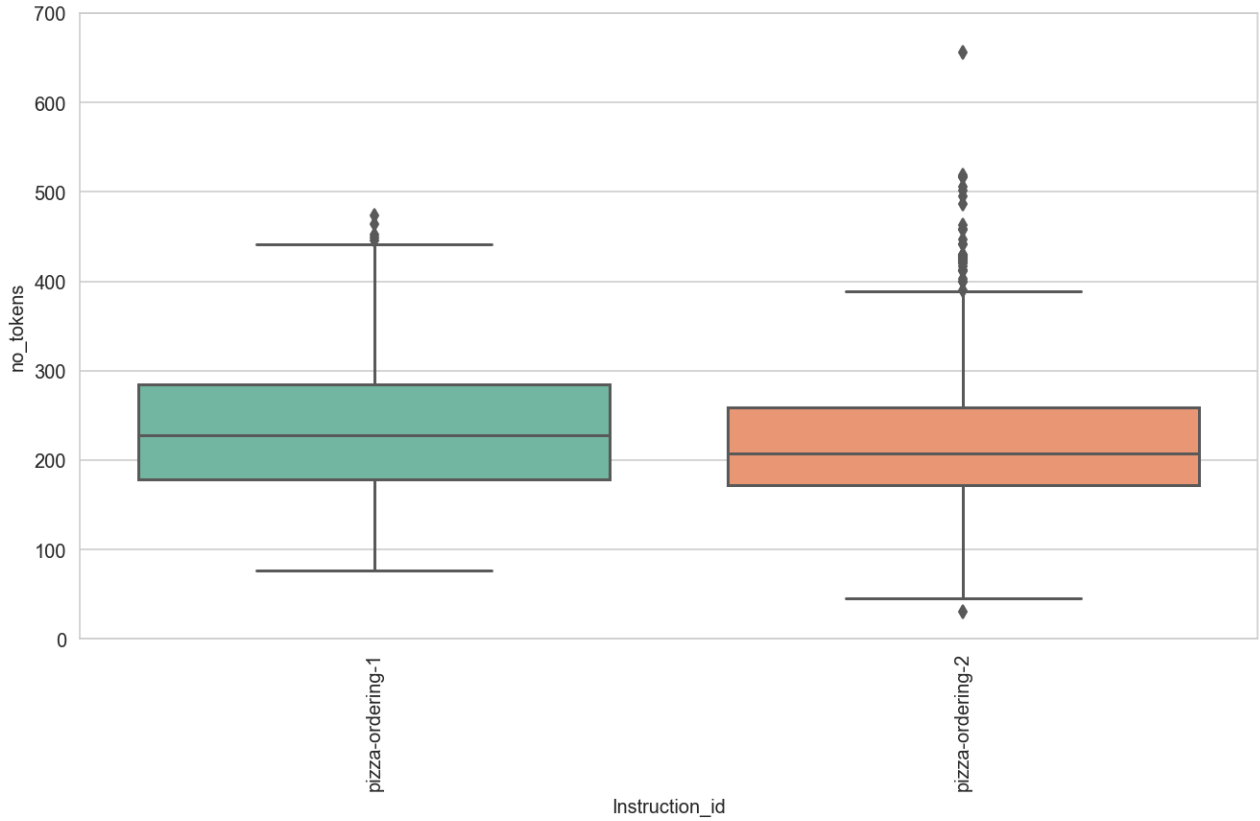
Cleaning and NLP

Data Exploration

- Statistical Analysis
- Exploring Word Clouds
- **Exploring Complexity**

Feature Engineering & Selection

Model Evaluation & Selection



Exploring Complexity

- Reviewed for service type: average number of tokens; distribution of tokens; printed outliers; compared # of tokens per ‘instruction_id’

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Balanced Dataset

Features Extracted:

- Counting Methods
  - Bag of Words
  - Bag of N-Grams
  - PCA on Bag of Words
  - Combined Features: BOW + Bag of N-Grams
  - TF-IDF
- Word Embeddings
  - Word2Vec
  - Word2Vec from FastText
  - Glove from FLAIR

Feature Selection Method:

- Univariate – SelectKBest "Chi2'
- PCA

Model Evaluation & Selection

Feature Extraction and Selection Summary

Vector Types	Feature Extracted	Scaling	Feature Selection Method
Count Vectors	Bag-of-words	MinMaxScaler()	Univariate Chi²
	Bag of n-grams	MinMaxScaler()	Univariate Chi²
	Bag-of-words	MinMaxScaler()	PCA
	Bag-of-words + Bag of n-grams	MinMaxScaler()	Univariate Chi²
	TF-IDF	MaxAbsScaler()	Univariate Chi²
Word Embeddings	Word2Vec from Word2Vec model	MinMaxScaler()	Univariate Chi²
	Word2Vec from FastText model	MinMaxScaler()	Univariate Chi²
	GloVe Embeddings with Flair	MinMaxScaler()	Univariate Chi²

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Balanced Dataset

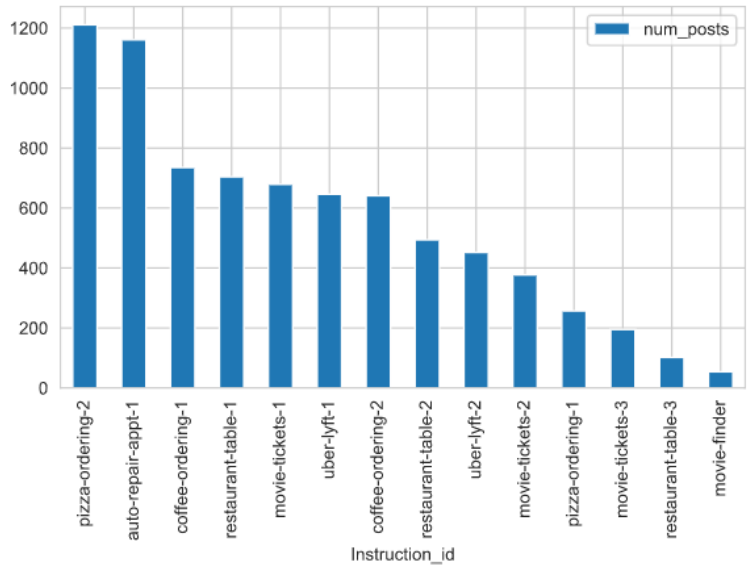
Features Extracted:

- Counting Methods
  - Bag of Words
  - Bag of N-Grams
  - PCA on Bag of Words
  - Combined Features: BOW + Bag of N-Grams
  - TF-IDF
- Word Embeddings
  - Word2Vec
  - Word2Vec from FastText
  - Glove from FLAIR

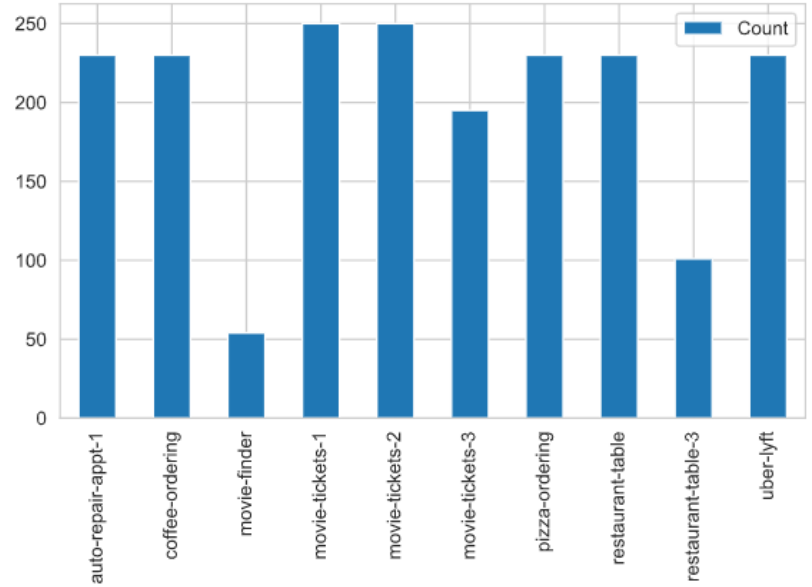
Feature Selection Method:

- Univariate – SelectKBest “Chi2’
- PCA

Model Evaluation & Selection



Original Dataset: Unbalanced



Balanced Dataset

Data Balancing:

- The original dataset was not balanced, the steps to balance were:
  - Merging of similar classes
  - Manual setting of number of samples

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Balanced Dataset

Features Extracted:

- **Counting Methods**
  - **Bag of Words**
  - Bag of N-Grams
  - PCA on Bag of Words
  - Combined Features: BOW + Bag of N-Grams
  - TF-IDF
- Word Embeddings
  - Word2Vec
  - Word2Vec from FastText
  - Glove from FLAIR

Feature Selection Method:

- **Univariate – SelectKBest "Chi2'**
- PCA

Model Evaluation & Selection

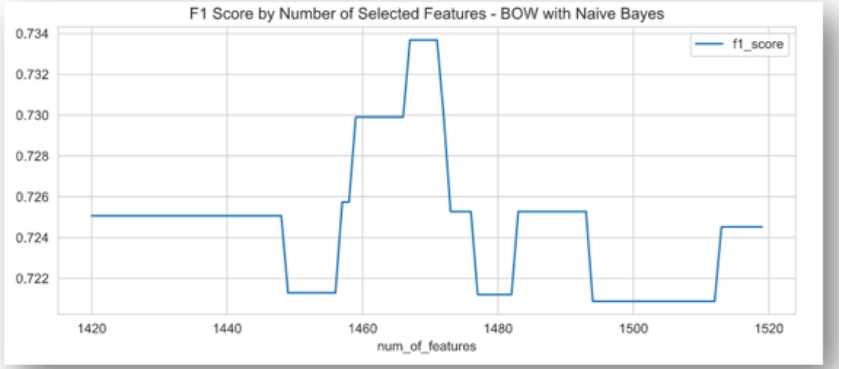
Bag of Words

Feature Extraction

	PAD	like	would	ok	okay	yes	want	pm	order	thank	time	tickets
0	0	3	3	3	0	1	1	1	0	0	0	0
1	0	0	0	3	0	0	2	0	0	0	1	0
2	0	1	0	0	2	3	1	5	0	2	0	5
3	0	0	0	0	3	0	1	5	0	1	3	0
4	0	2	1	0	2	2	2	0	3	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
995	0	5	6	0	4	2	2	0	6	2	1	0
996	0	3	3	0	1	1	1	0	0	0	0	0
997	0	1	0	5	0	3	0	0	0	4	0	0
998	0	0	0	0	5	1	1	4	0	0	1	1
999	0	6	5	0	4	2	0	8	1	0	2	0

1000 rows × 6115 columns

Feature Selection – Univariate with Chi²



Benchmarking

	Features_Benchedmarked	Feat_Type	Precision	Recall	f1_score	accuracy
0	BOW Naive Bayes Baseline	BOW	0.7243102	0.6960000	0.6838492	0.6960000
1	BOW Naive Bayes Optimal Features Selected: 1470	BOW	0.7656818	0.7360000	0.7336865	0.7360000

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Balanced Dataset

Features Extracted:

- **Counting Methods**
  - Bag of Words
  - **Bag of N-Grams**
  - PCA on Bag of Words
  - Combined Features: BOW + Bag of N-Grams
  - TF-IDF
- Word Embeddings
  - Word2Vec
  - Word2Vec from FastText
  - Glove from FLAIR

Feature Selection Method:

- **Univariate – SelectKBest "Chi2'**
- PCA

Model Evaluation & Selection

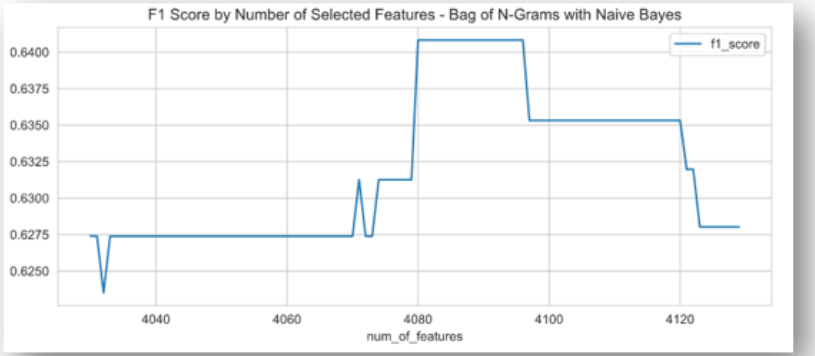
Bag of n-Grams

Feature Extraction

	abbey drive	abigail lives	abigails whoops	abigails yes	ability scan	able accommodate	able attend	able come	able find	able get	able make	able meet
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
745	0	0	0	0	0	0	0	0	0	0	0	0
746	0	0	0	0	0	0	0	0	0	0	0	0
747	0	0	0	0	0	0	0	0	0	0	0	0
748	0	0	0	0	0	0	0	0	0	0	0	0
749	0	0	0	0	0	0	0	0	0	0	0	0

750 rows × 37534 columns

Feature Selection – Univariate with Chi^2



Benchmarking

	Features_Benchedmarked	Feat_Type	Precision	Recall	f1_score	accuracy
0	Bag of N-Gram Naive Bayes baseline	BONG	0.6401657	0.6000000	0.5697283	0.6000000
1	Bag of N-Gram Naive Bayes Optimal Features Selected: 4080	BONG	0.6797066	0.6480000	0.6408318	0.6480000

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Balanced Dataset

Features Extracted:

- **Counting Methods**
  - Bag of Words
  - Bag of N-Grams
  - **PCA on Bag of Words**
  - Combined Features: BOW + Bag of N-Grams
  - TF-IDF
- Word Embeddings
  - Word2Vec
  - Word2Vec from FastText
  - Glove from FLAIR

Feature Selection Method:

- Univariate – SelectKBest "Chi2'
- **PCA**

Model Evaluation & Selection

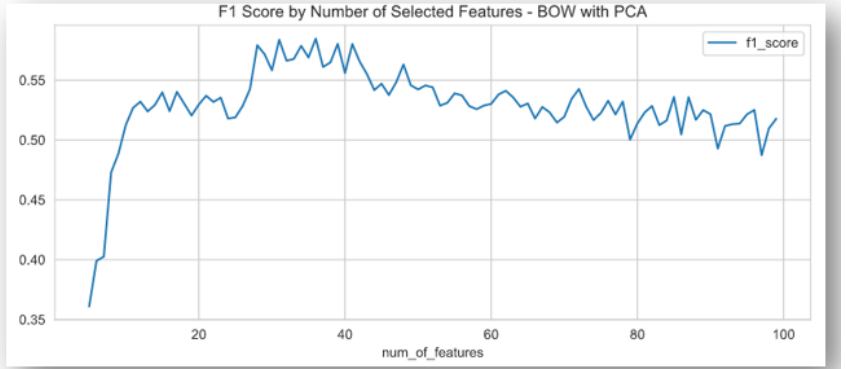
PCA on Bag of Words

Feature Extraction

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	1.37	-1.61	-0.65	0.40	-2.47	-1.83	0.66	0.93	0.35	0.45	1.39	1.58	-0.88
1	-2.02	-2.62	0.33	-1.64	-0.99	-0.85	-2.02	0.87	2.24	-2.80	-0.42	0.22	-0.51
2	-3.09	3.23	1.07	-2.81	0.11	-0.45	-0.94	-0.00	-0.34	0.01	-1.27	0.09	-2.37
3	2.66	0.69	-2.27	2.91	-3.35	3.07	-1.85	-1.83	2.40	2.49	1.14	-0.56	-0.33
4	-2.38	-4.20	-1.04	0.02	-0.68	0.32	-0.51	-0.28	-0.03	-1.49	0.60	-0.13	0.46
...	...	...	...	...	...	...	...	...	...	...	...	...	...
745	-3.07	-0.57	0.11	-3.86	-0.67	-1.92	-1.83	0.87	1.02	0.05	-0.16	-0.69	0.69
746	-1.78	-1.48	2.56	1.75	-0.34	3.39	0.92	3.44	0.29	-1.81	-0.19	-0.40	0.18
747	-3.72	1.31	-4.59	-3.17	2.58	0.59	4.80	-1.17	0.02	2.34	0.15	0.58	-0.36
748	-1.90	-4.91	-2.57	-0.83	0.13	1.19	2.24	-2.61	0.83	0.23	0.04	-2.68	1.55
749	0.58	5.87	-5.46	-4.87	0.16	0.72	3.34	-0.25	1.99	3.47	0.84	-0.67	1.92

750 rows × 38 columns

Feature Selection – PCA



Benchmarking

	Features_Benchedmarked	Feat_Type	Precision	Recall	f1_score	accuracy
1	BOW With Top: 36 PCA Components Seleted	BOW_PCA	0.6207596	0.5920000	0.5804592	0.5920000

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Balanced Dataset

Features Extracted:

- **Counting Methods**
  - Bag of Words
  - Bag of N-Grams
  - PCA on Bag of Words
  - **Combined Features: BOW + Bag of N-Grams**
  - TF-IDF
- Word Embeddings
  - Word2Vec
  - Word2Vec from FastText
  - Glove from FLAIR

Feature Selection Method:

- **Univariate – SelectKBest "Chi2'**
- PCA

Model Evaluation & Selection

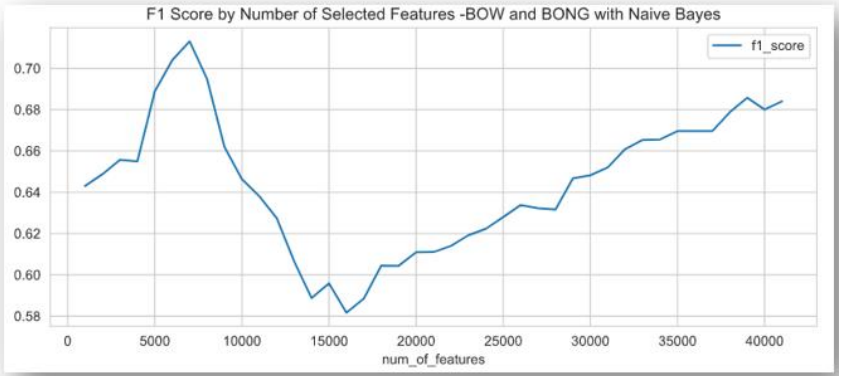
Combination – BOW and Bag of nGrams

Feature Extraction

	0	1	2	3	4	5
0	0.0000000	5.0000000	4.0000000	0.0000000	0.0000000	5.0000000
1	0.0000000	1.0000000	3.0000000	0.0000000	0.0000000	3.0000000
2	0.0000000	1.0000000	1.0000000	3.0000000	0.0000000	2.0000000
3	0.0000000	4.0000000	4.0000000	0.0000000	9.0000000	1.0000000
4	0.0000000	0.0000000	1.0000000	6.0000000	0.0000000	0.0000000
...	...	...	...	...	...	...
245	0.0000000	0.0000000	0.0000000	0.0000000	7.0000000	4.0000000
246	0.0000000	5.0000000	2.0000000	0.0000000	0.0000000	4.0000000
247	0.0000000	1.0000000	0.0000000	0.0000000	2.0000000	3.0000000
248	0.0000000	1.0000000	1.0000000	1.0000000	0.0000000	0.0000000
249	0.0000000	0.0000000	0.0000000	0.0000000	2.0000000	3.0000000

250 rows × 43649 columns

Feature Selection – Univariate with Chi^2



Benchmarking

	Features_Benchedmarked	Feat_Type	Precision	Recall	f1_score	accuracy
0	BOW + Bag of NGrams Top: 7000 Features with Naive Bayes	BOW_BONG	0.7848327	0.7200000	0.7130762	0.7200000

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Balanced Dataset

Features Extracted:

- **Counting Methods**
  - Bag of Words
  - Bag of N-Grams
  - PCA on Bag of Words
  - Combined Features: BOW + Bag of N-Grams
  - **TF-IDF**
- Word Embeddings
  - Word2Vec
  - Word2Vec from FastText
  - Glove from FLAIR

Feature Selection Method:

- **Univariate – SelectKBest "Chi2'**
- PCA

Model Evaluation & Selection

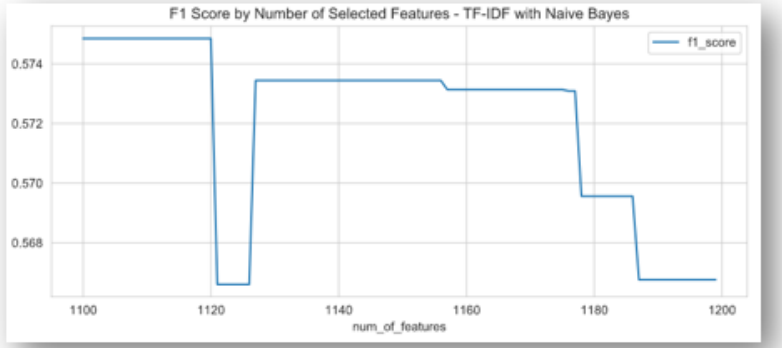
TF-IDF

Feature Extraction

	abbey	abigail	abigails	ability	able	abosolutly	abotu
0	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
1	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
2	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
3	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
4	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
...	...	...	...	...	...	...	...
745	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
746	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
747	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
748	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
749	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000

750 rows × 5221 columns

Feature Selection – Univariate with Chi²



Benchmarking

	Features_Benchedmarked	Feat_Type	Precision	Recall	f1_score	accuracy
0	TF-IDF Naive Bayes Baseline	TF-IDF	0.6391206	0.6080000	0.5563710	0.6080000
1	TF-IDF Naive Bayes Optimal Features Selected: 1100	TF-IDF	0.7315170	0.6280000	0.5748536	0.6280000

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Balanced Dataset

Features Extracted:

- Counting Methods
  - Bag of Words
  - Bag of N-Grams
  - PCA on Bag of Words
  - Combined Features: BOW + Bag of N-Grams
  - TF-IDF
- **Word Embeddings**
  - **Word2Vec**
  - Word2Vec from FastText
  - Glove from FLAIR

Feature Selection Method:

- **Univariate – SelectKBest "Chi2'**
- PCA

Model Evaluation & Selection

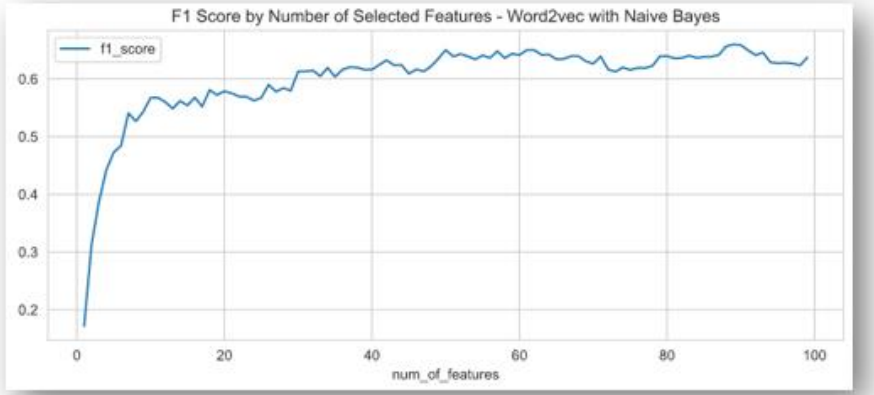
Word2Vec

Feature Extraction

	0	1	2	3	4	5
0	0.1485421	-0.7538016	-0.1134053	0.6662195	-0.7246261	-0.7991774
1	-0.0607553	1.0541956	0.4737130	-0.1523951	0.8053292	-0.4724531
2	1.0977465	0.2787353	0.1757089	0.1846928	0.1721655	-0.5292627
3	0.5198169	0.1662873	-0.9000216	-0.3820015	-0.5178834	0.2152452
4	0.9221550	0.6896073	-0.1255608	-0.7041550	-1.0570785	-0.2343613
...	...	...	...	...	...	...
745	0.7777841	0.2974014	0.0622340	0.0823392	0.1991102	-0.0638373
746	1.3327909	1.0493979	0.4724289	-0.1673032	-0.3365210	-0.0422698
747	1.1111339	0.1478708	0.4795337	0.4620506	0.4769705	-0.6561098
748	0.7482316	-0.0639104	1.0775760	0.2059366	-0.5521994	-0.7394230
749	1.1768541	-0.0455626	-0.4162890	0.0041162	0.6534415	-0.8599626

750 rows × 100 columns

Feature Selection – Univariate with Chi²



Benchmarking

	Features_Benchedmarked	Feat_Type	Precision	Recall	f1_score	accuracy
0	Word2Vec Naive Bayes Baseline	Word2Vec	0.6512414	0.6480000	0.6379862	0.6480000
1	Word2Vec Naive Bayes Optimal Features Selected: 89	Word2Vec	0.6792014	0.6640000	0.6595766	0.6640000

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Balanced Dataset

Features Extracted:

- Counting Methods
  - Bag of Words
  - Bag of N-Grams
  - PCA on Bag of Words
  - Combined Features: BOW + Bag of N-Grams
  - TF-IDF
- **Word Embeddings**
  - Word2Vec
  - **Word2Vec from FastText**
  - Glove from FLAIR

Feature Selection Method:

- **Univariate – SelectKBest "Chi2'**
- PCA

Model Evaluation & Selection

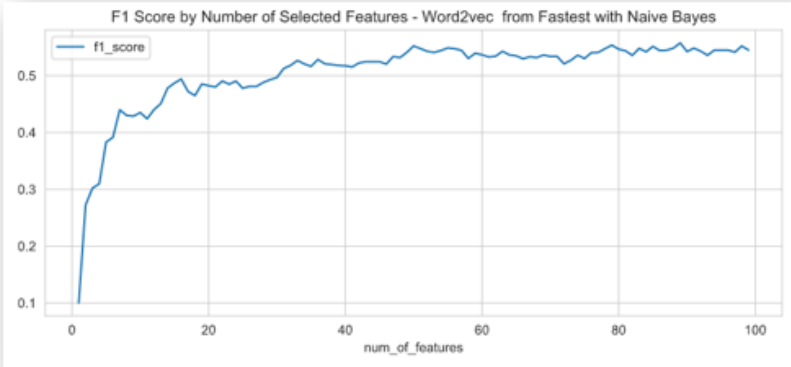
Word2Vec from FastText

Feature Extraction

	0	1	2	3	4	5
0	0.1485421	-0.7538016	-0.1134053	0.6662195	-0.7246261	-0.7991774
1	-0.0607553	1.0541956	0.4737130	-0.1523951	0.8053292	-0.4724531
2	1.0977465	0.2787353	0.1757089	0.1846928	0.1721655	-0.5292627
3	0.5198169	0.1662873	-0.9000216	-0.3820015	-0.5178834	0.2152452
4	0.9221550	0.6896073	-0.1255608	-0.7041550	-1.0570785	-0.2343613
...	...	...	...	...	...	...
745	0.7777841	0.2974014	0.0622340	0.0823392	0.1991102	-0.0638373
746	1.3327909	1.0493979	0.4724289	-0.1673032	-0.3365210	-0.0422698
747	1.1111339	0.1478708	0.4795337	0.4620506	0.4769705	-0.6561098
748	0.7482316	-0.0639104	1.0775760	0.2059366	-0.5521994	-0.7394230
749	1.1768541	-0.0455626	-0.4162890	0.0041162	0.6534415	-0.8599626

750 rows × 100 columns

Feature Selection – Univariate with Chi^2



Benchmarking

	Features_Benchedmarked	Feat_Type	Precision	Recall	f1_score	accuracy
0	Word2Vec Fastext Naive Bayes Baseline	Word2Vec_FT	0.6491872	0.5960000	0.5457620	0.5960000
1	Word2Vec from Fastest Naive Bayes Optimal Features Selected: 89	Word2Vec_FT	0.6491047	0.6080000	0.5575331	0.6080000

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Balanced Dataset

Features Extracted:

- Counting Methods
  - Bag of Words
  - Bag of N-Grams
  - PCA on Bag of Words
  - Combined Features: BOW + Bag of N-Grams
  - TF-IDF
- **Word Embeddings**
  - Word2Vec
  - Word2Vec from FastText
  - **Glove from FLAIR**

Feature Selection Method:

- **Univariate – SelectKBest "Chi2'**
- PCA

Model Evaluation & Selection

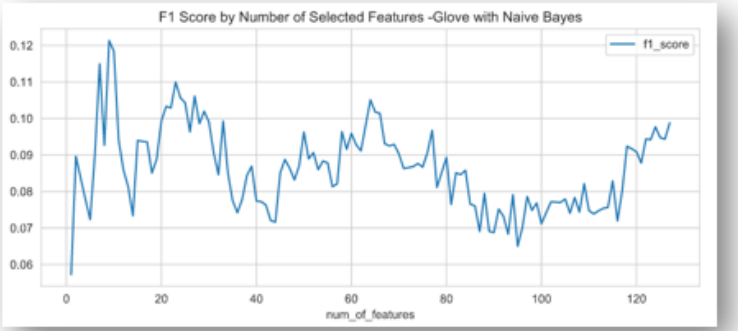
Glove from FLAIR

Feature Extraction

	0	1	2	3	4	5	6
0	-0.0117013	0.0617747	-0.0950755	0.1091563	-0.1017631	0.1596720	-0.0589565
1	0.0841807	-0.0620297	-0.1189875	-0.2875261	-0.0688366	0.0544598	-0.0740813
2	0.5025941	-0.0467734	-0.0588523	0.0163325	-0.1757609	-0.0487810	-0.0511439
3	-0.0227185	-0.0103282	-0.2404889	-0.1926294	-0.0662659	0.0504382	-0.2372182
4	-0.3222020	0.1958944	-0.4945650	-0.0861031	-0.0720596	0.1695689	-0.1488497
...	...	...	...	...	...	...	...
245	0.1313466	-0.2277337	-0.0699241	-0.1615932	-0.0721090	-0.0488684	-0.0867414
246	-0.2333377	-0.0675329	-0.3608953	-0.0170508	-0.2104710	0.1508308	-0.1324217
247	-0.1406810	0.1325267	-0.1478698	-0.1698386	-0.2022910	0.0391697	0.0031608
248	-0.1477484	-0.1279918	-0.3581616	-0.1365404	-0.1992545	-0.0465445	-0.2704552
249	0.4757498	-0.0056710	-0.0204031	0.0933946	-0.2478866	-0.0336119	0.0204238

250 rows × 128 columns

Feature Selection – Univariate with Chi²



Benchmarking

	Features_Benchedmarked	f1_score	accuracy
0	Glove with Naive Bayes All Features	0.13	0.14

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Balanced Dataset

Features Extracted:

- Counting Methods
  - Bag of Words
  - Bag of N-Grams
  - PCA on Bag of Words
  - Combined Features: BOW + Bag of N-Grams
  - TF-IDF
- Word Embeddings
  - Word2Vec
  - Word2Vec from FastText
  - Glove from FLAIR

Feature Selection Method:

- Univariate – SelectKBest “Chi2’
- PCA

Model Evaluation & Selection

Feature Engineering, Extraction and Selection Final Results

	Features_Benchedmarked	Feat_Type	Precision	Recall	f1_score	accuracy
0	BOW Naive Bayes Baseline	BOW	0.7243102	0.6960000	0.6838492	0.6960000
1	BOW Naive Bayes Optimal Features Selected: 1470	BOW	0.7656818	0.7360000	0.7336865	0.7360000
2	Bag of N-Gram Naive Bayes baseline	BONG	0.6401657	0.6000000	0.5697283	0.6000000
3	Bag of N-Gram Naive Bayes Optimal Features Selected: 4080	BONG	0.6797066	0.6480000	0.6408318	0.6480000
4	TF-IDF Naive Bayes Baseline	TF-IDF	0.6391206	0.6080000	0.5563710	0.6080000
5	TF-IDF Naive Bayes Optimal Features Selected: 1100	TF-IDF	0.7315170	0.6280000	0.5748536	0.6280000
6	Word2Vec Naive Bayes Baseline	Word2Vec	0.6512414	0.6480000	0.6379862	0.6480000
7	Word2Vec Naive Bayes Optimal Features Selected: 89	Word2Vec	0.6792014	0.6640000	0.6595766	0.6640000
8	Word2Vec Fastext Naive Bayes Baseline	Word2Vec_FT	0.6491872	0.5960000	0.5457620	0.5960000
9	Word2Vec from Fastest Naive Bayes Optimal Features Selected: 89	Word2Vec_FT	0.6491047	0.6080000	0.5575331	0.6080000
10	BOW + Bag of NGrams Top: 7000 Features with Naive Bayes	BOW_BONG	0.7848327	0.7200000	0.7130762	0.7200000
11	BOW and Bag of N-Grams Combined Baseline	BOW_BONG	0.7848327	0.7200000	0.7130762	0.7200000
12	BOW With Top: 36 PCA Components Seleted	BOW_PCA	0.6207596	0.5920000	0.5804592	0.5920000

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Model Evaluation & Selection

Features Selected

- Bag of Words
- Univariate SelectKBest - chi2

Models Benchmark

- Logistic Regression
- Multinomial Naïve Bayes
- Random Forest Classifier
- Linear SVC

Optimized Parameters

- GridSearchCV

Ensemble Learning

- Bagging
- Boosting
- Stacking
- Voting

Modeling, Model Evaluation & Tuning Summary

Step	Description
1. Benchmark • **Using Bag of Words features** • **Feature Selection using Chi²**	Logistic Regression
	Multinomial Naïve Bayes
	Random Forest Classifier
	Linear SVC
2. Optimized Hyperparameters	GridSearchCV
3. Learning Curves	Accuracy
	Errors
4. ROC Curves	AUC
5. Ensemble Learning	Bagging
	Boosting
	Stacking
	Voting

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Model Evaluation & Selection

Features Selected

- **Bag of Words**
- **Univariate SelectKBest - chi2**

Models Benchmark

- Logistic Regression
- Multinomial Naïve Bayes
- Random Forest Classifier
- Linear SVC

Optimized Parameters

- GridSearchCV

Ensemble Learning

- Bagging
- Boosting
- Stacking
- Voting

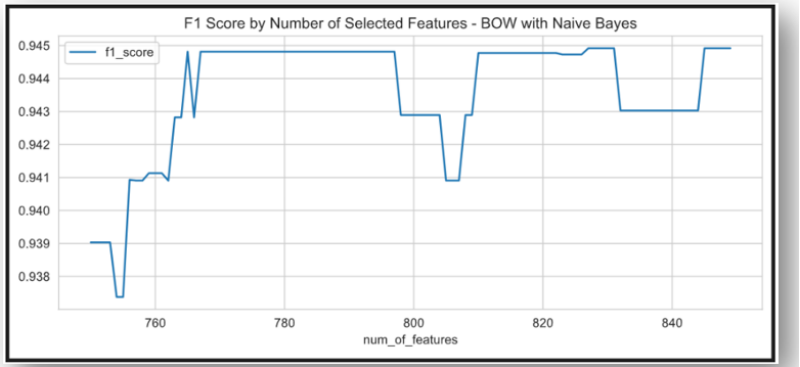
Best Features Selected - Bag of Words

Feature Extraction

	PAD	like	would	ok	okay	yes	want	pm	order	thank	time	tickets
0	0	3	3	3	0	1	1	1	0	0	0	0
1	0	0	0	3	0	0	2	0	0	0	1	0
2	0	1	0	0	2	3	1	5	0	2	0	5
3	0	0	0	0	3	0	1	5	0	1	3	0
4	0	2	1	0	2	2	2	0	3	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
995	0	5	6	0	4	2	2	0	6	2	1	0
996	0	3	3	0	1	1	1	0	0	0	0	0
997	0	1	0	5	0	3	0	0	0	4	0	0
998	0	0	0	0	5	1	1	4	0	0	1	1
999	0	6	5	0	4	2	0	8	1	0	2	0

1000 rows × 6115 columns

Feature Selection – Univariate with Chi^2



Benchmarking

	Features_Benchedmarked	Feat_Type	Precision	Recall	f1_score	accuracy
0	BOW Naive Bayes Baseline	BOW	0.9306527	0.9240000	0.9253405	0.9240000
1	BOW Naive Bayes Optimal Features Selected: 849	BOW	0.9478845	0.9440000	0.9449159	0.9440000

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Model Evaluation & Selection

Features Selected

- Bag of Words
- Univariate SelectKBest - chi2

Models Benchmark

- **Logistic Regression**
- **Multinomial Naïve Bayes**
- **Random Forest Classifier**
- **Linear SVC**

Optimized Parameters

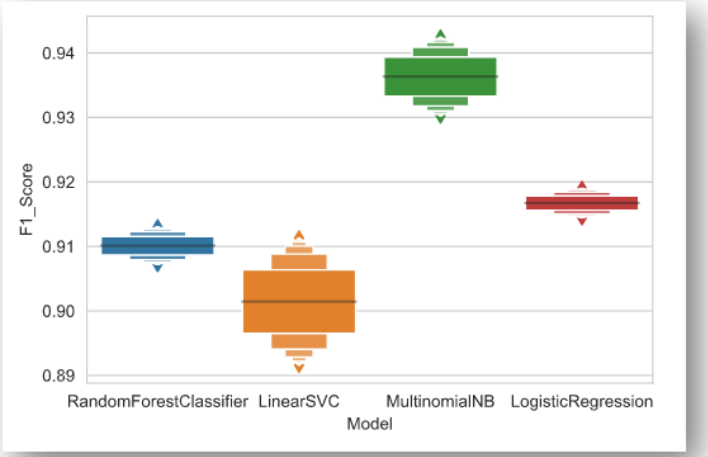
- GridSearchCV

Ensemble Learning

- Bagging
- Boosting
- Stacking
- Voting

Models Baseline Benchmark Comparison

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
0	RF	RandomForestClassifier	baseline	default		0.9072162
1	SVC	LinearSVC	baseline	default		0.9112999
2	NB	MultinomialNB	baseline	default		0.9302587
3	LR	LogisticRegression	baseline	default		0.9144730
4	RF	RandomForestClassifier	optimized	default		0.9130000
5	SVC	LinearSVC	optimized	default		0.8915947
6	NB	MultinomialNB	optimized	default		0.9424083
7	LR	LogisticRegression	optimized	default		0.9189740



- **Baseline**: Bag of Words Features
- **Optimized**: Bag of Words using Univariate Chi^2 feature selection method

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Model Evaluation & Selection

Features Selected

- Bag of Words
- Univariate SelectKBest - chi2

Models Benchmark

- Logistic Regression
- Multinomial Naïve Bayes
- Random Forest Classifier
- Linear SVC

Optimized Parameters

- **GridSearchCV**

Ensemble Learning

- Bagging
- Boosting
- Stacking
- Voting

Optimized Hyperparameters Using GridSearchCV

```python
from sklearn.model_selection import GridSearchCV

class Estimator_Parameters:
    def __init__(self, estimator, parameters, feat_type, x, y):
        self.estimator = estimator
        self.parameters = parameters
        self.feat_type = feat_type
        self.x = x
        self.y = y

def Get_Best_Parameters(est_param):
    grid_search = GridSearchCV(estimator = est_param.estimator,
                               param_grid = est_param.parameters,
                               scoring = 'f1_weighted',
                               cv= 10,
                               n_jobs = -1)
    grid_search = grid_search.fit(est_param.x, est_param.y)
    return grid_search.best_score_, grid_search.best_params_
```

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
0	RF	RandomForestClassifier	baseline	default		0.9072162
1	SVC	LinearSVC	baseline	default		0.9112999
2	NB	MultinomialNB	baseline	default		0.9302587
3	LR	LogisticRegression	baseline	default		0.9144730
4	RF	RandomForestClassifier	optimized	default		0.9130000
5	SVC	LinearSVC	optimized	default		0.8915947
6	NB	MultinomialNB	optimized	default		0.9424083
7	LR	LogisticRegression	optimized	default		0.9189740
8	RF	RandomForestClassifier	optimized	tuned	{'max_depth': 6, 'n_estimators': 90, 'random_state': 2}	0.8968076
9	SVC	LinearSVC	optimized	tuned	{'C': 1300, 'dual': False, 'loss': 'squared_hinge', 'max_iter': 1100, 'penalty': 'l1'}	0.9067355
10	NB	MultinomialNB	optimized	tuned	{'alpha': 0.3, 'fit_prior': False}	0.9429960
11	LR	LogisticRegression	optimized	tuned	{'C': 0.1, 'dual': False, 'multi_class': 'auto', 'penalty': 'l2'}	0.9202755

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Model Evaluation & Selection

Features Selected

- Bag of Words
- Univariate SelectKBest - chi2

Models Benchmark

- Logistic Regression
- Multinomial Naïve Bayes
- Random Forest Classifier
- Linear SVC

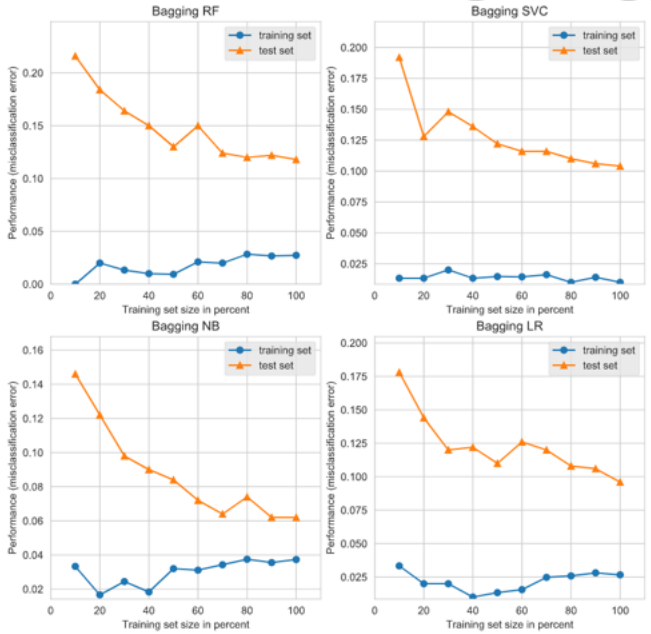
Optimized Parameters

- GridSearchCV

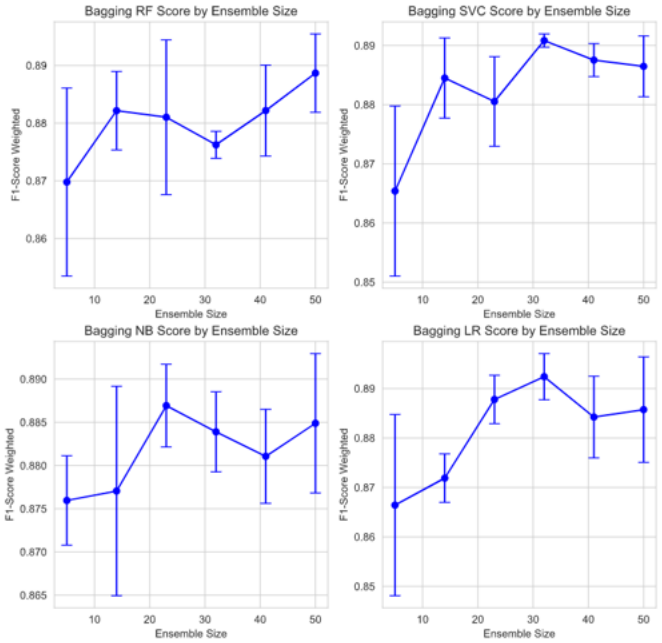
Ensemble Learning

- **Bagging**
- Boosting
- Stacking
- Voting

Ensemble Learning – Bagging



Learning Curves Test/Train Errors



F1-Score Weighted by Ensemble Size

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
12	RF	Bagging RF	optimized	tuned		0.8889733
13	SVC	Bagging SVC	optimized	tuned		0.9246471
14	NB	Bagging NB	optimized	tuned		0.9372484
15	LR	Bagging LR	optimized	tuned		0.9103760

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Model Evaluation & Selection

Features Selected

- Bag of Words
- Univariate SelectKBest - chi2

Models Benchmark

- Logistic Regression
- Multinomial Naïve Bayes
- Random Forest Classifier
- Linear SVC

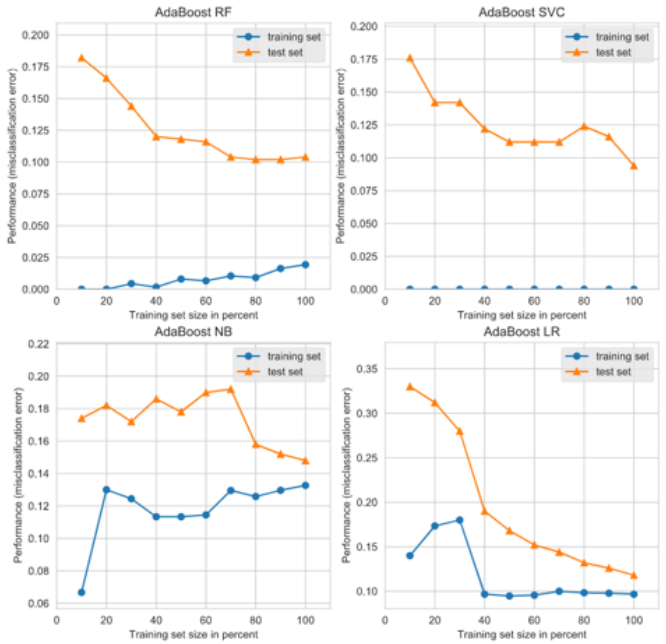
Optimized Parameters

- GridSearchCV

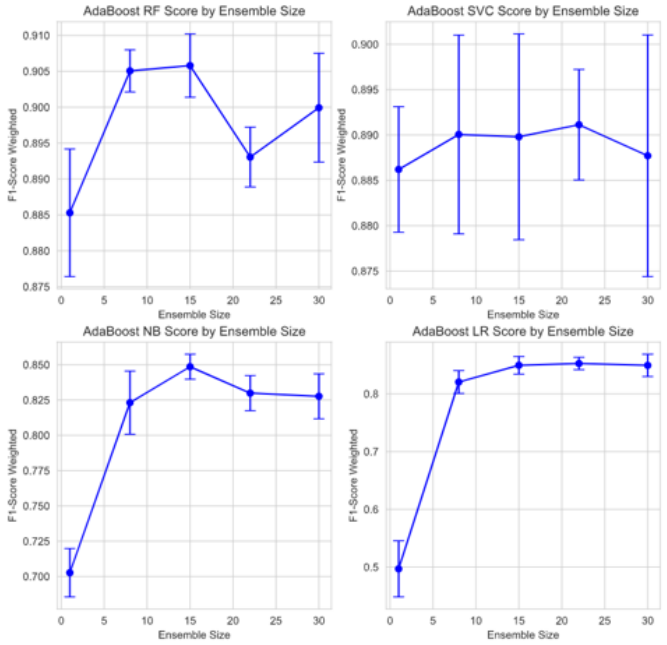
Ensemble Learning

- Bagging
- **Boosting**
- Stacking
- Voting

Ensemble Learning – Boosting



Learning Curves Test/Train Errors



F1-Score Weighted by Ensemble Size

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
16	RF	AdaBoost RF	optimized	tuned		0.9040332
17	SVC	AdaBoost SVC	optimized	tuned		0.8915750
18	NB	AdaBoost NB	optimized	tuned		0.8485944
19	LR	AdaBoost LR	optimized	tuned		0.8526345

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Model Evaluation & Selection

Features Selected

- Bag of Words
- Univariate SelectKBest - chi2

Models Benchmark

- Logistic Regression
- Multinomial Naïve Bayes
- Random Forest Classifier
- Linear SVC

Optimized Parameters

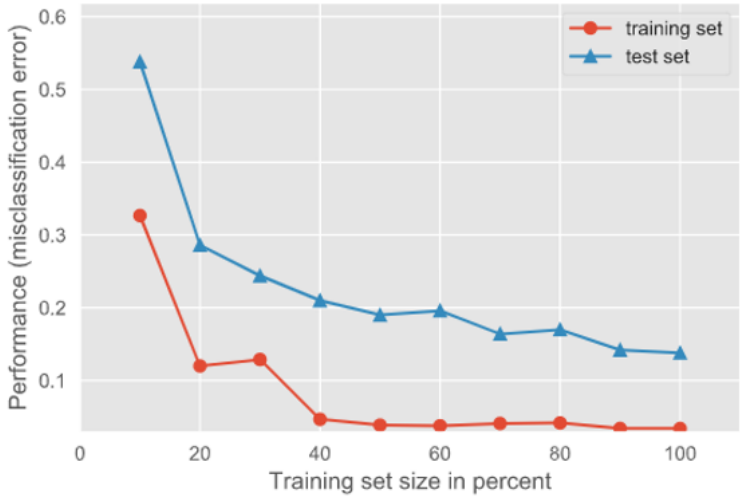
- GridSearchCV

Ensemble Learning

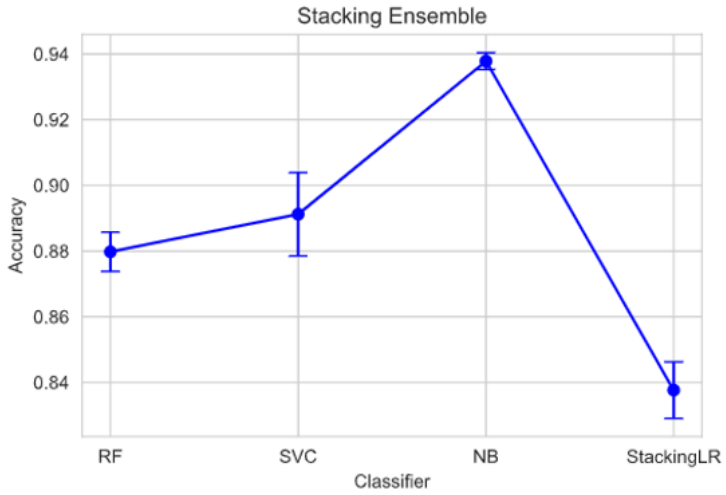
- Bagging
- Boosting
- **Stacking**
- Voting

Ensemble Learning – Stacking

Random Forest -> Linear SVC -> Multinomial Naïve Bayes -> **Stacking Logistic Regression**



Learning Curves Test/Train Errors



Stacking Ensemble Accuracy by Classifier

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
20	Stack	Stacking LR	optimized	tuned		0.8376481
21	Stack	Stacking Boosted LR	optimized	tuned		0.8213541

Data Preparation

Cleaning and NLP

Data Exploration

Feature Engineering & Selection

Model Evaluation & Selection

Features Selected

- Bag of Words
- Univariate SelectKBest - chi2

Models Benchmark

- Logistic Regression
- Multinomial Naïve Bayes
- Random Forest Classifier
- Linear SVC

Optimized Parameters

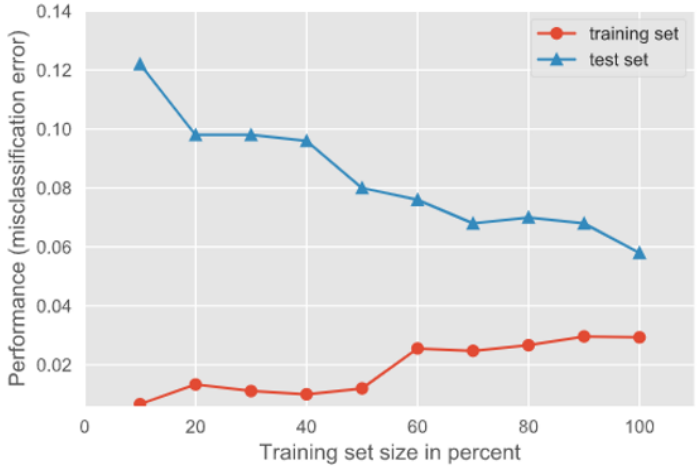
- GridSearchCV

Ensemble Learning

- Bagging
- Boosting
- Stacking
- **Voting**

Ensemble Learning - Voting

```
from sklearn.ensemble import VotingClassifier

clf2 = svm.SVC(kernel='linear', probability=True)

labels = ['hard', 'soft']

for label in labels:
    eclf1 = VotingClassifier(estimators=[('Random Forest', clf1), ('MultinomialNB', clf3), ('Logistic Regr
ession', clf4)], voting=label)
    scores = cross_val_score(eclf1, rm_chi_opt_bow.x_train_sel, y_train, cv=3, scoring='f1_weighted')
    print("Accuracy: %.7f (+/- %.2f) [%s]" %(scores.mean(), scores.std(), "Voting: " + label))
    results = Result_Update_Or_Append('Voting', 'Voting ' + label, 'optimized', 'tuned', '', scores.mean
(), False)

eclf2 = VotingClassifier(estimators=[('Random Forest', clf1), ('LinearSVC', clf2), ('MultinomialNB', clf
3), ('Logistic Regression', clf4)], voting='soft', weights=[1.5, 0.7,1.5,0.9], flatten_transform=True)
scores = cross_val_score(eclf2, rm_chi_opt_bow.x_train_sel, y_train, cv=3, scoring='f1_weighted')
print("Accuracy: %.7f (+/- %.2f) [%s]" %(scores.mean(), scores.std(), "Weighted Voting: " + label))
results = Result_Update_Or_Append('Voting', 'Weighted Voting soft', 'optimized', 'tuned', '', scores.mean
(), False)

cv_df = pd.DataFrame(results, columns=['Model_Id','Model', 'Features', 'Hyper_Param', 'Best_Params', 'F1_S
core'])
```



	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
22	Voting	Voting hard	optimized	tuned		0.9323097
23	Voting	Voting soft	optimized	tuned		0.9396819
24	Voting	Weighted Voting soft	optimized	tuned		0.9416539

1. **Baseline Models Benchmarks**
2. Feature Optimized Models Benchmarks
3. Hyperparameter Tuned Benchmarks
   - Learning Curves
   - ROC Curves / AUC
4. Bagging Ensemble Learning Benchmarks
   - Training / Testing Errors
   - F1-Score by Ensemble Size
5. Boosting Ensemble Learning Benchmarks
   - Training / Testing Errors
   - F1-Score by Ensemble Size
6. Stacking Ensemble Learning Benchmarks
   - Training / Testing Errors
   - F1-Score by Classifier
7. Voting Ensemble Learning Benchmarks
   - Training / Testing Errors
8. Overall Benchmarks

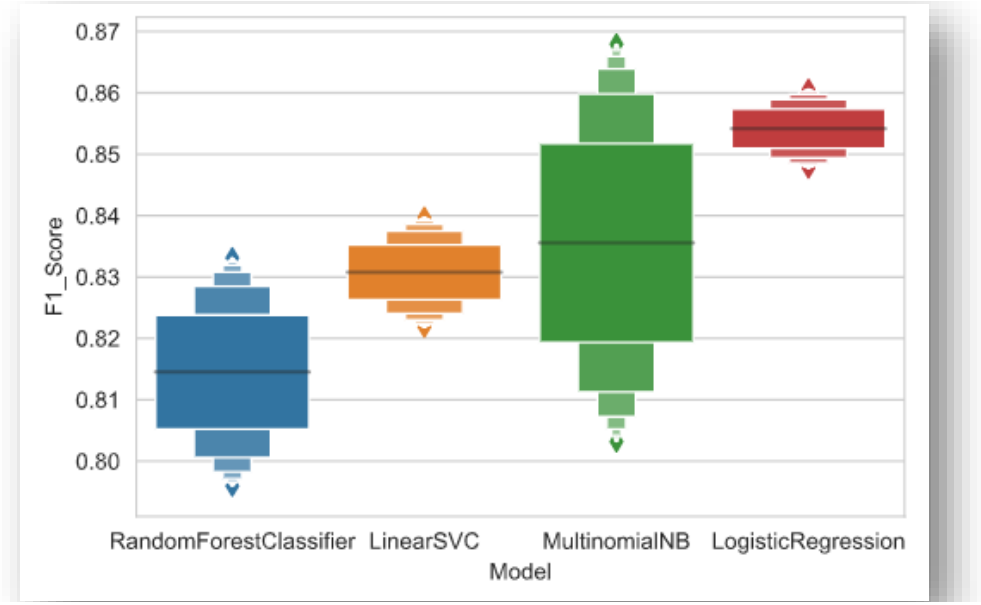
	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
0	RF	RandomForestClassifier	baseline	default		0.7959575
1	SVC	LinearSVC	baseline	default		0.8397106
2	NB	MultinomialNB	baseline	default		0.8031807
3	LR	LogisticRegression	baseline	default		0.8605552

Baseline Models Benchmarks

1. Baseline Models Benchmarks
2. **Feature Optimized Models Benchmarks**
3. Hyperparameter Tuned Benchmarks
   - Learning Curves
   - ROC Curves / AUC
4. Bagging Ensemble Learning Benchmarks
   - Training / Testing Errors
   - F1-Score by Ensemble Size
5. Boosting Ensemble Learning Benchmarks
   - Training / Testing Errors
   - F1-Score by Ensemble Size
6. Stacking Ensemble Learning Benchmarks
   - Training / Testing Errors
   - F1-Score by Classifier
7. Voting Ensemble Learning Benchmarks
   - Training / Testing Errors
8. Overall Benchmarks

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
4	RF	RandomForestClassifier	optimized	default		0.8331067
5	SVC	LinearSVC	optimized	default		0.8218430
6	NB	MultinomialNB	optimized	default		0.8679468
7	LR	LogisticRegression	optimized	default		0.8477885

Feature Optimized Models Benchmarks

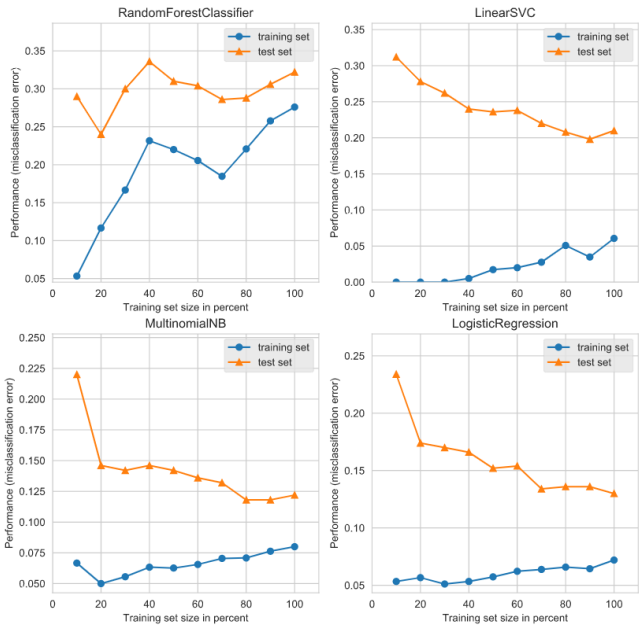


Comparison of Baseline and Feature Optimized Models Benchmarks.

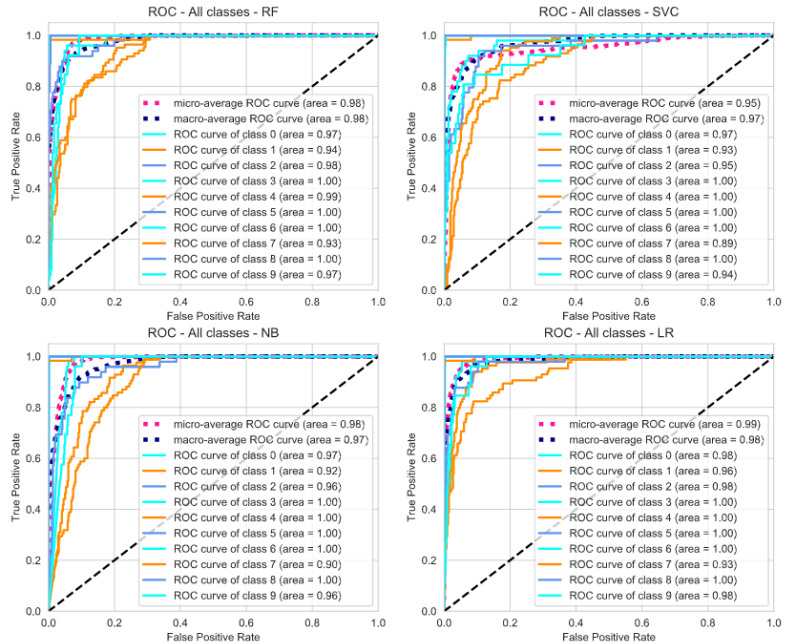
1. Baseline Models Benchmarks
2. Feature Optimized Models Benchmarks
3. **Hyperparameter Tuned Benchmarks**
   - **Learning Curves**
   - **ROC Curves / AUC**
4. Bagging Ensemble Learning Benchmarks
   - Training / Testing Errors
   - F1-Score by Ensemble Size
5. Boosting Ensemble Learning Benchmarks
   - Training / Testing Errors
   - F1-Score by Ensemble Size
6. Stacking Ensemble Learning Benchmarks
   - Training / Testing Errors
   - F1-Score by Classifier
7. Voting Ensemble Learning Benchmarks
   - Training / Testing Errors
8. Overall Benchmarks

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
8	RF	RandomForestClassifier	optimized	tuned	{'max_depth': 6, 'n_estimators': 90, 'random_state': 0}	0.6289337
9	SVC	LinearSVC	optimized	tuned	{'C': 1500, 'dual': True, 'loss': 'squared_hinge', 'max_iter': 1100, 'penalty': 'l2'}	0.7802828
10	NB	MultinomialNB	optimized	tuned	{'alpha': 0.44, 'fit_prior': False}	0.8794765
11	LR	LogisticRegression	optimized	tuned	{'C': 0.1, 'dual': False, 'multi_class': 'auto', 'penalty': 'l2'}	0.8544467

Hyperparameters Tuned Benchmarks



Training / Testing Errors

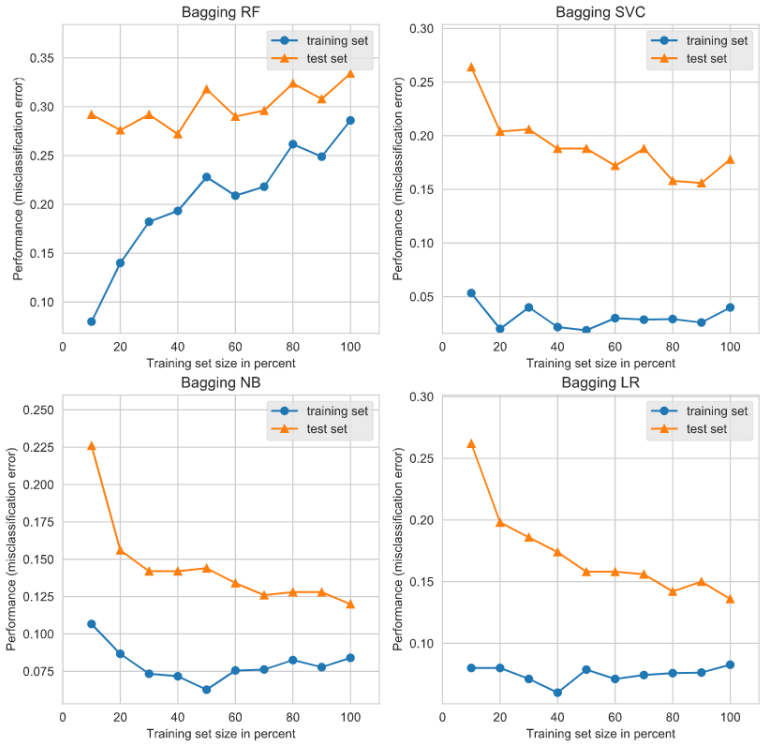


ROC Curves / AUC

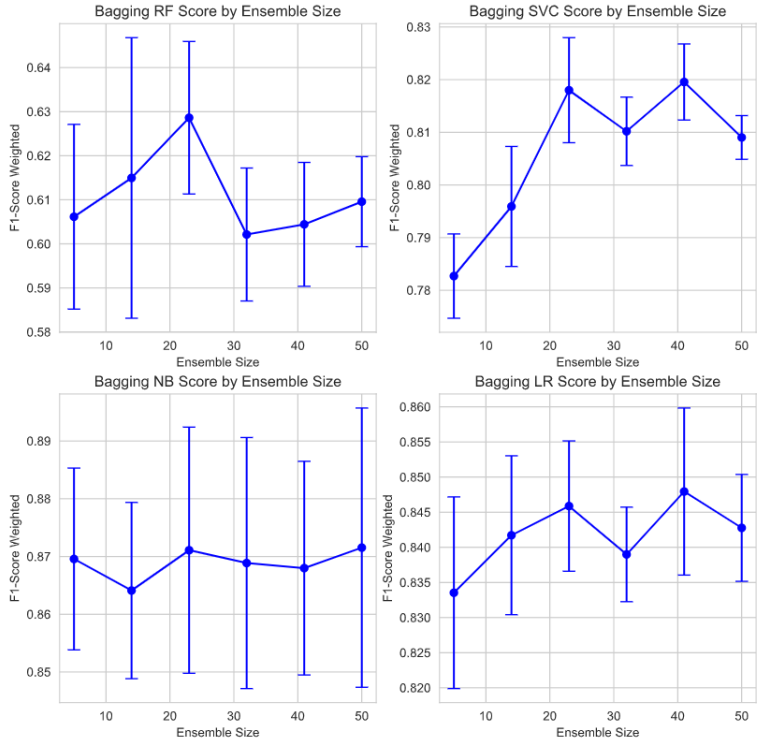
1. Baseline Models Benchmarks
2. Feature Optimized Models Benchmarks
3. Hyperparameter Tuned Benchmarks
   - Learning Curves
   - ROC Curves / AUC
4. **Bagging Ensemble Learning Benchmarks**
   - **Training / Testing Errors**
   - **F1-Score by Ensemble Size**
5. Boosting Ensemble Learning Benchmarks
   - Training / Testing Errors
   - F1-Score by Ensemble Size
6. Stacking Ensemble Learning Benchmarks
   - Training / Testing Errors
   - F1-Score by Classifier
7. Voting Ensemble Learning Benchmarks
   - Training / Testing Errors
8. Overall Benchmarks

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
12	RF	Bagging RF	optimized	tuned		0.6082327
13	SVC	Bagging SVC	optimized	tuned		0.8007875
14	NB	Bagging NB	optimized	tuned		0.8698887
15	LR	Bagging LR	optimized	tuned		0.8314868

Bagging Ensemble Learning Benchmarks



Training / Testing Errors

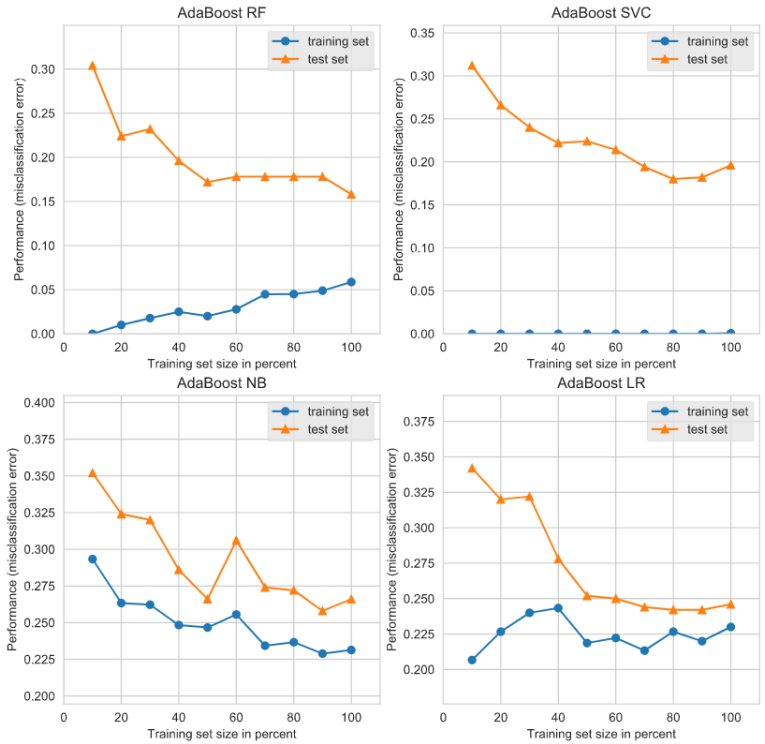


F1-Score By Ensemble Size

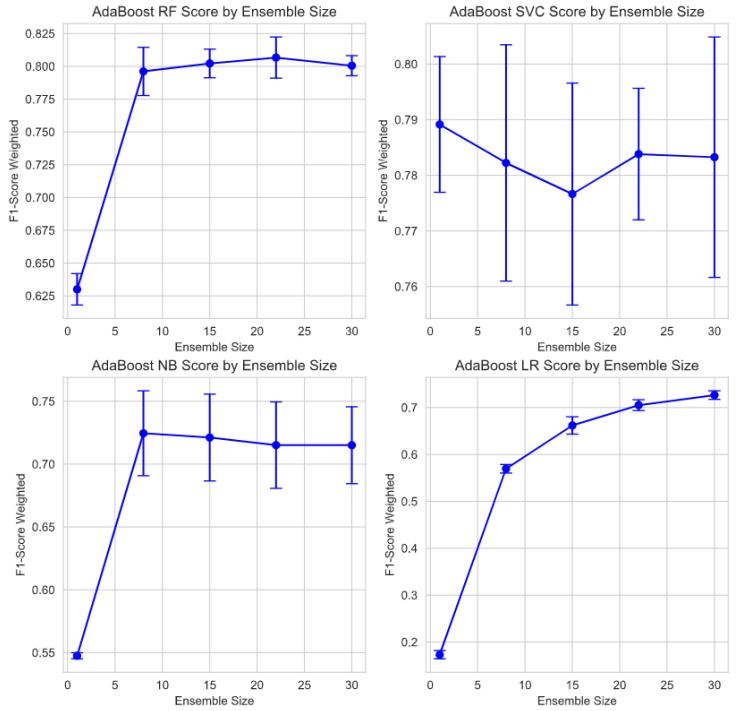
1. Baseline Models Benchmarks
2. Feature Optimized Models Benchmarks
3. Hyperparameter Tuned Benchmarks
   - Learning Curves
   - ROC Curves / AUC
4. Bagging Ensemble Learning Benchmarks
   - Training / Testing Errors
   - F1-Score by Ensemble Size
5. **Boosting Ensemble Learning Benchmarks**
   - **Training / Testing Errors**
   - **F1-Score by Ensemble Size**
6. Stacking Ensemble Learning Benchmarks
   - Training / Testing Errors
   - F1-Score by Classifier
7. Voting Ensemble Learning Benchmarks
   - Training / Testing Errors
8. Overall Benchmarks

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
16	RF	AdaBoost RF	optimized	tuned		0.7962695
17	SVC	AdaBoost SVC	optimized	tuned		0.7898456
18	NB	AdaBoost NB	optimized	tuned		0.7245388
19	LR	AdaBoost LR	optimized	tuned		0.7265426

Boosting Ensemble Learning Benchmarks



Training / Testing Errors



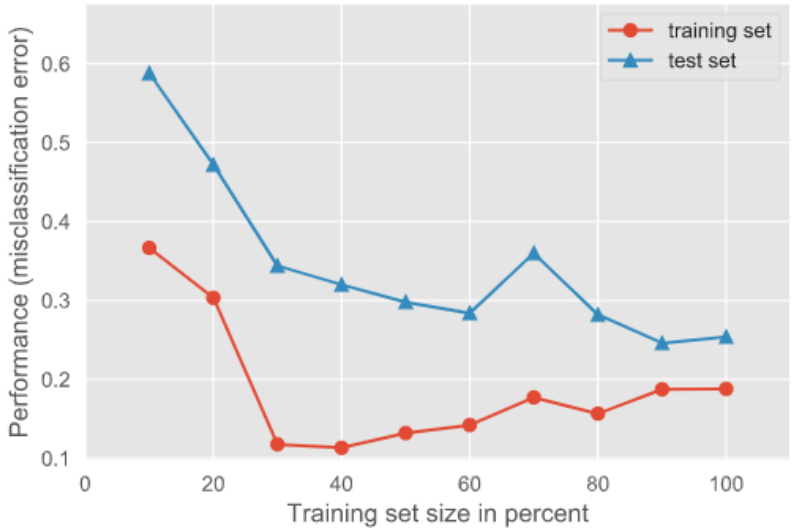
F1-Score By Ensemble Size

1. Baseline Models Benchmarks
2. Feature Optimized Models Benchmarks
3. Hyperparameter Tuned Benchmarks
   - Learning Curves
   - ROC Curves / AUC
4. Bagging Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
5. Boosting Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
6. **Stacking Ensemble Learning Benchmarks**
   - **Training / Testing Errors**
   - **F1-Score by Classifier**
7. Voting Ensemble Learning Benchmarks
   - Training / Testing Errors
8. Overall Benchmarks

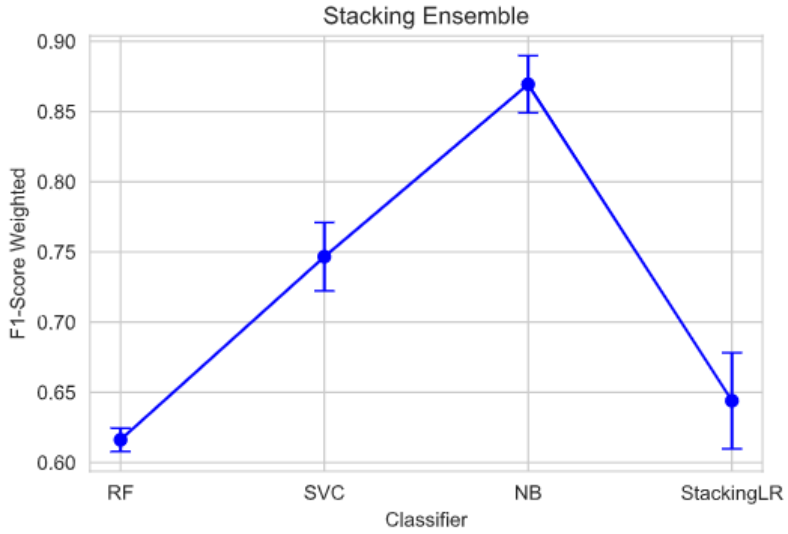
	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
20	Stack	Stacking LR	optimized	tuned		0.6440204
21	Stack	Stacking Boosted LR	optimized	tuned		0.4591639

Stacking Ensemble Learning Benchmarks

Random Forest -> Linear SVC -> Multinomial Naïve Bayes -> **Stacking Logistic Regression**



Training / Testing Errors

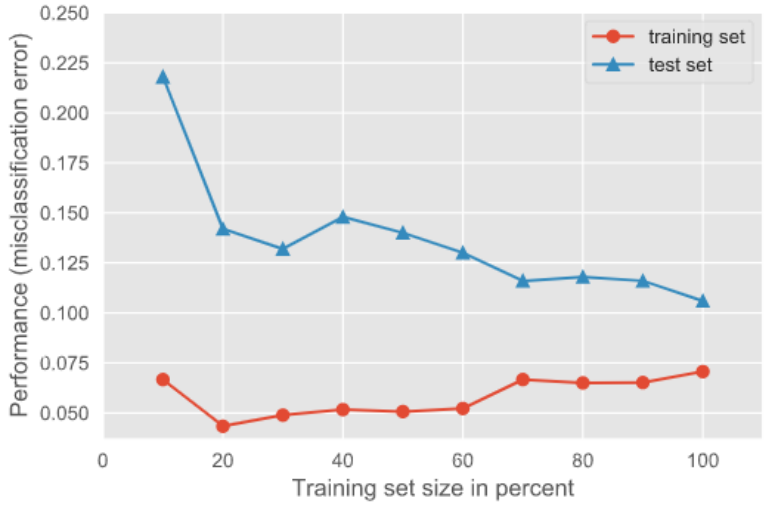


F1-Score By Classifier

1. Baseline Models Benchmarks
2. Feature Optimized Models Benchmarks
3. Hyperparameter Tuned Benchmarks
   - Learning Curves
   - ROC Curves / AUC
4. Bagging Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
5. Boosting Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
6. Stacking Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
7. **Voting Ensemble Learning Benchmarks**
   - **Training / Testing Errors**
8. Overall Benchmarks

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
22	Voting	Voting hard	optimized	tuned		0.8492384
23	Voting	Voting soft	optimized	tuned		0.8680407
24	Voting	Weighted Voting soft	optimized	tuned		0.8692584

Voting Ensemble Learning Benchmarks



Training / Testing Errors

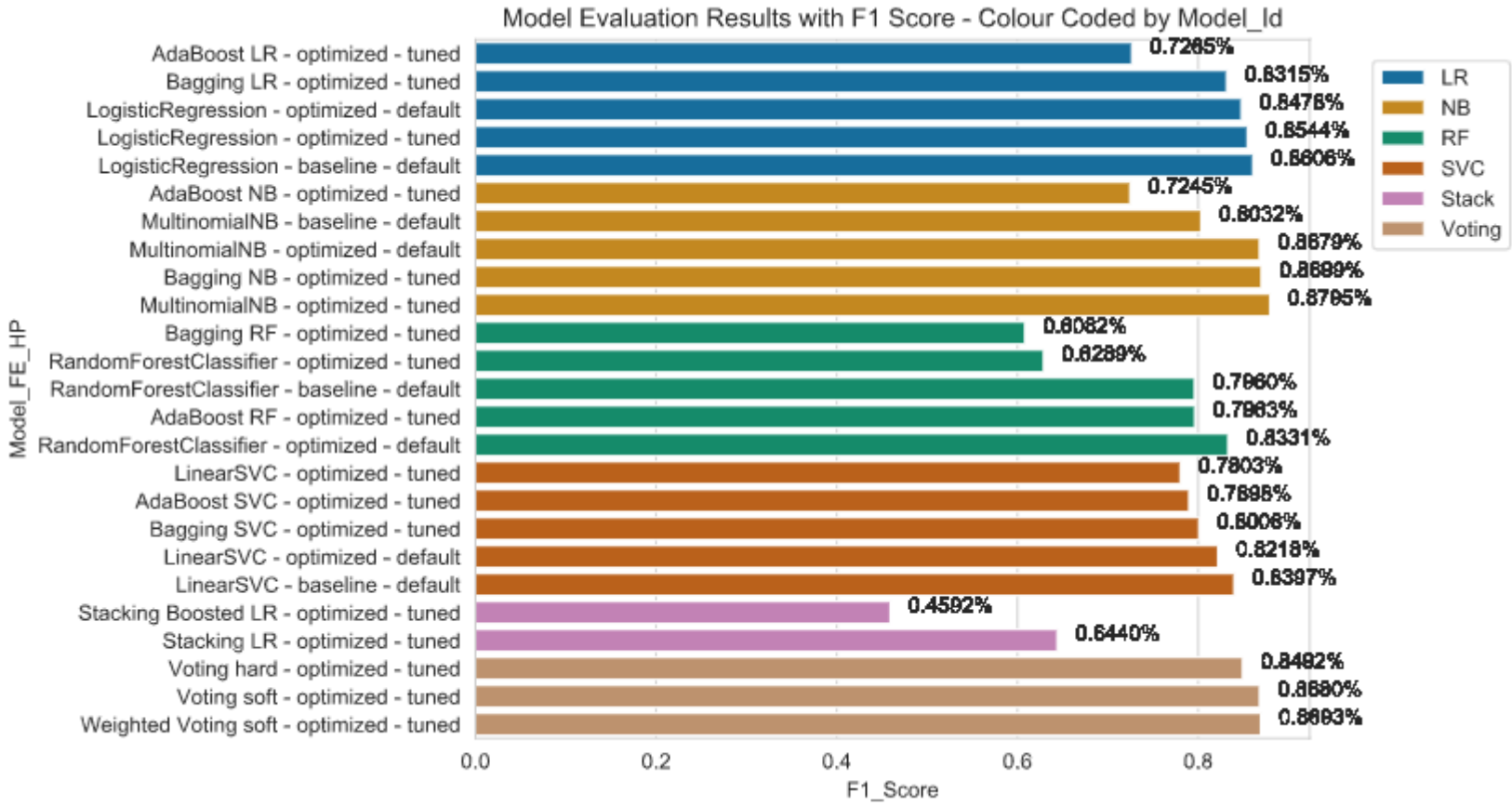
Voting Classifier supports two types of voting:

- **hard:** the final class prediction is made by a majority vote — the estimator chooses the class prediction that occurs most frequently among the base models
- **soft:** the final class prediction is made based on the average probability calculated using all the base model predictions.

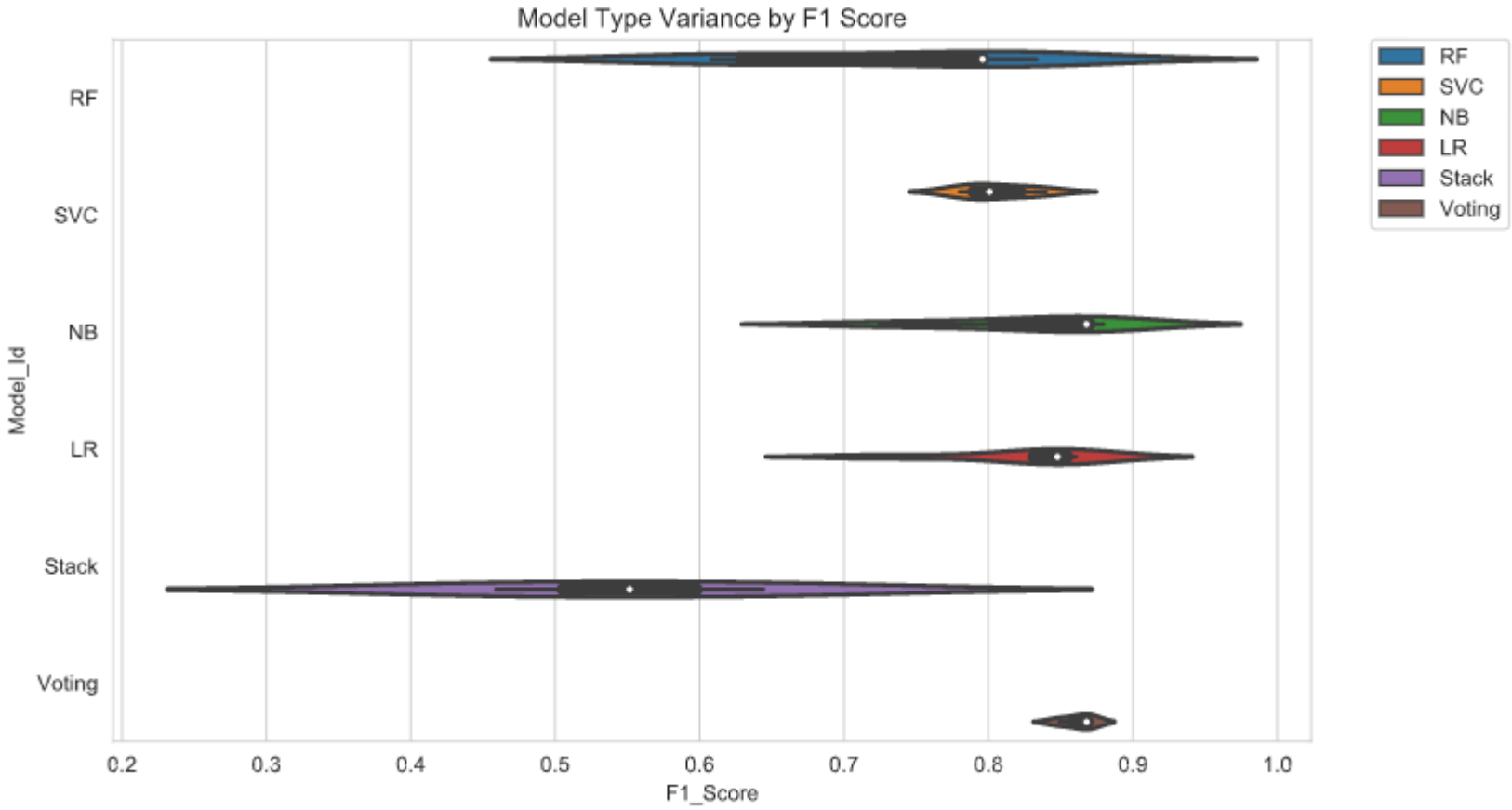
1. Baseline Models Benchmarks
2. Feature Optimized Models Benchmarks
3. Hyperparameter Tuned Benchmarks
   - Learning Curves
   - ROC Curves / AUC
4. Bagging Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
5. Boosting Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
6. Stacking Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
7. Voting Ensemble Learning Benchmarks
   - Training / Testing Errors
8. **Overall Benchmarks**

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
0	RF	RandomForestClassifier	baseline	default		0.7959575
1	SVC	LinearSVC	baseline	default		0.8397106
2	NB	MultinomialNB	baseline	default		0.8031807
3	LR	LogisticRegression	baseline	default		0.8605552
4	RF	RandomForestClassifier	optimized	default		0.8331067
5	SVC	LinearSVC	optimized	default		0.8218430
6	NB	MultinomialNB	optimized	default		0.8679468
7	LR	LogisticRegression	optimized	default		0.8477885
8	RF	RandomForestClassifier	optimized	tuned	{'max_depth': 6, 'n_estimators': 90, 'random_state': 0}	0.6289337
9	SVC	LinearSVC	optimized	tuned	{'C': 1500, 'dual': True, 'loss': 'squared_hinge', 'max_iter': 1100, 'penalty': 'l2'}	0.7802828
10	NB	MultinomialNB	optimized	tuned	{'alpha': 0.44, 'fit_prior': False}	0.8794765
11	LR	LogisticRegression	optimized	tuned	{'C': 0.1, 'dual': False, 'multi_class': 'auto', 'penalty': 'l2'}	0.8544467
12	RF	Bagging RF	optimized	tuned		0.6082327
13	SVC	Bagging SVC	optimized	tuned		0.8007875
14	NB	Bagging NB	optimized	tuned		0.8698887
15	LR	Bagging LR	optimized	tuned		0.8314868
16	RF	AdaBoost RF	optimized	tuned		0.7962695
17	SVC	AdaBoost SVC	optimized	tuned		0.7898456
18	NB	AdaBoost NB	optimized	tuned		0.7245388
19	LR	AdaBoost LR	optimized	tuned		0.7265426
20	Stack	Stacking LR	optimized	tuned		0.6440204
21	Stack	Stacking Boosted LR	optimized	tuned		0.4591639
22	Voting	Voting hard	optimized	tuned		0.8492384
23	Voting	Voting soft	optimized	tuned		0.8680407
24	Voting	Weighted Voting soft	optimized	tuned		0.8692584

1. Baseline Models Benchmarks
2. Feature Optimized Models Benchmarks
3. Hyperparameter Tuned Benchmarks
   - Learning Curves
   - ROC Curves / AUC
4. Bagging Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
5. Boosting Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
6. Stacking Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
7. Voting Ensemble Learning Benchmarks
   - Training / Testing Errors
8. **Overall Benchmarks**



1. Baseline Models Benchmarks
2. Feature Optimized Models Benchmarks
3. Hyperparameter Tuned Benchmarks
   - Learning Curves
   - ROC Curves / AUC
4. Bagging Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
5. Boosting Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
6. Stacking Ensemble Learning Benchmarks
   - Learning Curves
   - F1-Score by Ensemble Size
7. Voting Ensemble Learning Benchmarks
   - Training / Testing Errors
8. **Overall Benchmarks**

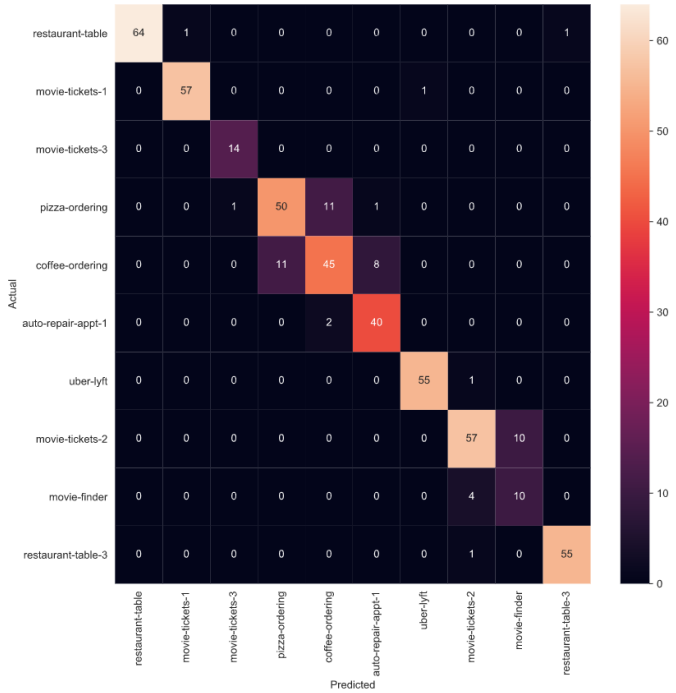


Comparison and Analysis

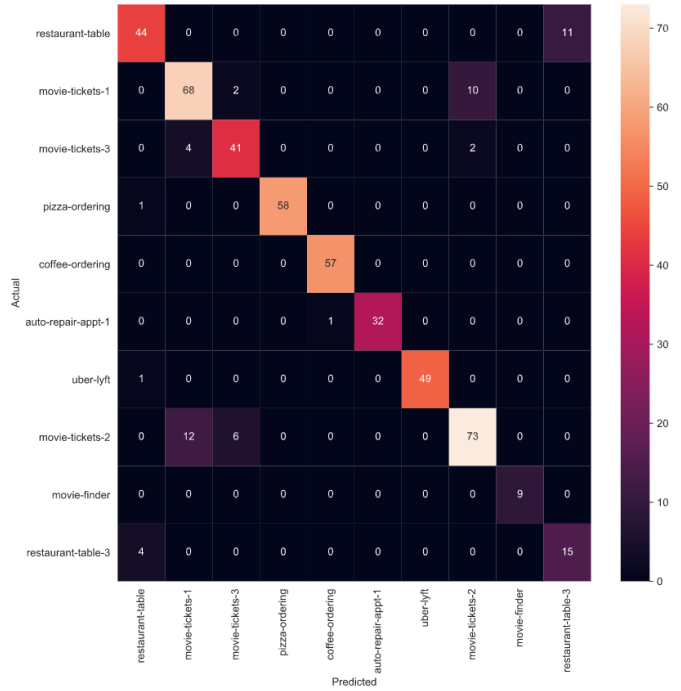
- **Confusion Matrix from M1 and M2**
- Comparison of the progression: from baseline thru ensemble
- Comparison of 2000 & 4000 records: Learning Curves and F1-Score (covers the bias-variance tradeoff)

Interpreting and Explaining Models

- Feature Importance
- Individual Tree Visualization
- ELI5
- LIME
- Skater



At End of Feature Selection



At the End of the Model Selection

Comparison of the confusion matrices at the end of feature selection and end of model evaluation and selection gives a visual of how the results improved.

Comparison and Analysis

- Confusion Matrix from M1 and M2
- **Comparison of the progression: from baseline thru ensemble**
- Comparison of 2000 & 4000 records: Learning Curves and F1-Score (covers the bias-variance tradeoff)

Interpreting and Explaining Models

- Feature Importance
- Individual Tree Visualization
- ELI5
- LIME
- Skater

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
0	RF	RandomForestClassifier	baseline	default		0.7959575
1	SVC	LinearSVC	baseline	default		0.8397106
2	NB	MultinomialNB	baseline	default		0.8031807
3	LR	LogisticRegression	baseline	default		0.8605552
4	RF	RandomForestClassifier	optimized	default		0.8331067
5	SVC	LinearSVC	optimized	default		0.8218430
6	NB	MultinomialNB	optimized	default		0.8679468
7	LR	LogisticRegression	optimized	default		0.8477885
8	RF	RandomForestClassifier	optimized	tuned	{'max_depth': 6, 'n_estimators': 90, 'random_state': 0}	0.6289337
9	SVC	LinearSVC	optimized	tuned	{'C': 1500, 'dual': True, 'loss': 'squared_hinge', 'max_iter': 1100, 'penalty': 'l2'}	0.7802828
10	NB	MultinomialNB	optimized	tuned	{'alpha': 0.44, 'fit_prior': False}	0.8794765
11	LR	LogisticRegression	optimized	tuned	{'C': 0.1, 'dual': False, 'multi_class': 'auto', 'penalty': 'l2'}	0.8544467
12	RF	Bagging RF	optimized	tuned		0.6082327
13	SVC	Bagging SVC	optimized	tuned		0.8007875
14	NB	Bagging NB	optimized	tuned		0.8698887
15	LR	Bagging LR	optimized	tuned		0.8314868
16	RF	AdaBoost RF	optimized	tuned		0.7962695
17	SVC	AdaBoost SVC	optimized	tuned		0.7898456
18	NB	AdaBoost NB	optimized	tuned		0.7245388
19	LR	AdaBoost LR	optimized	tuned		0.7265426
20	Stack	Stacking LR	optimized	tuned		0.6440204
21	Stack	Stacking Boosted LR	optimized	tuned		0.4591639
22	Voting	Voting hard	optimized	tuned		0.8492384
23	Voting	Voting soft	optimized	tuned		0.8680407
24	Voting	Weighted Voting soft	optimized	tuned		0.8692584

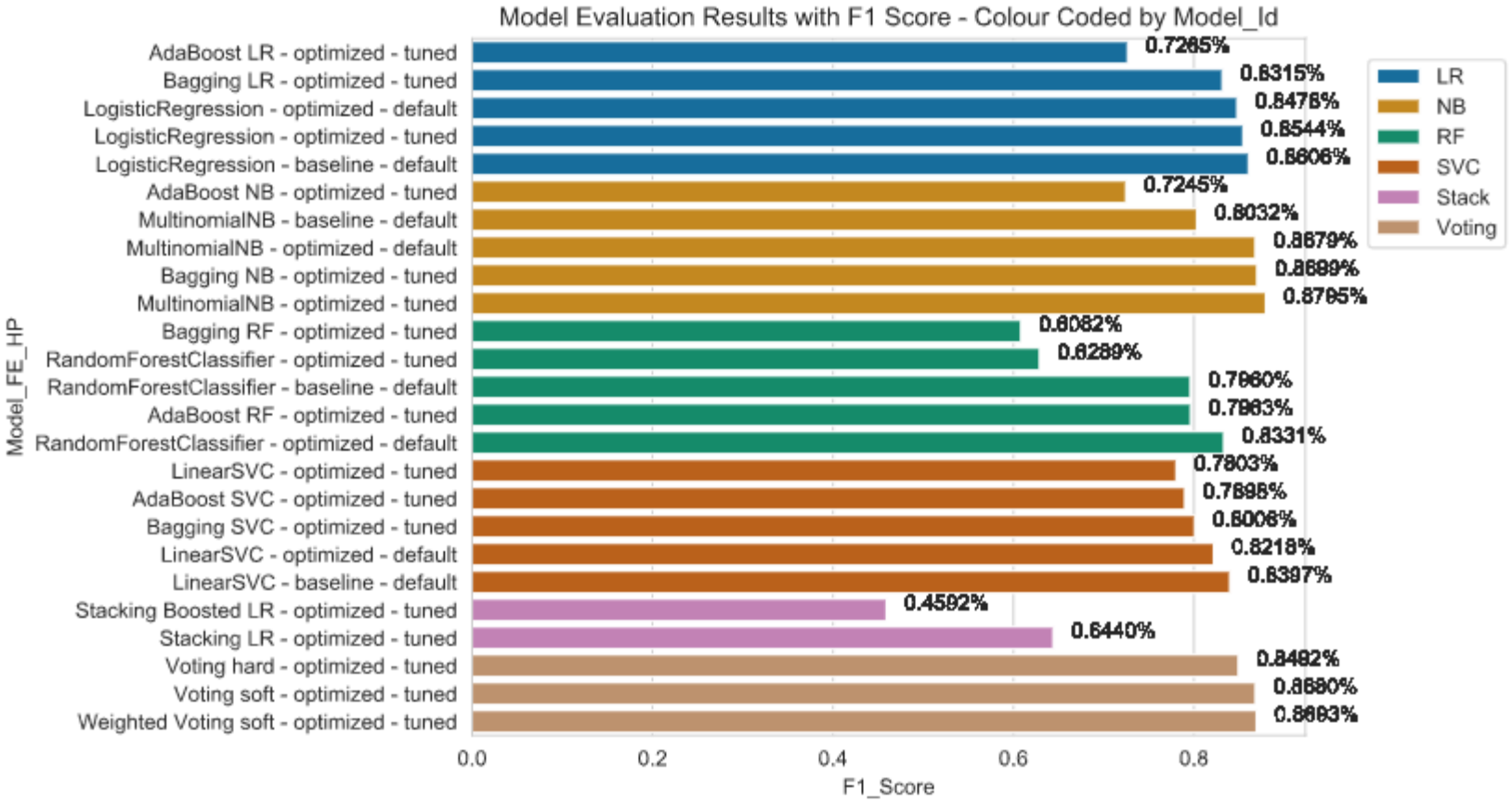
Comparison of the Progression from baseline thru ensemble learning

Comparison and Analysis

- Confusion Matrix from M1 and M2
- **Comparison of the progression: from baseline thru ensemble**
- Comparison of 2000 & 4000 records: Learning Curves and F1-Score (covers the bias-variance tradeoff)

Interpreting and Explaining Models

- Feature Importance
- Individual Tree Visualization
- ELI5
- LIME
- Skater



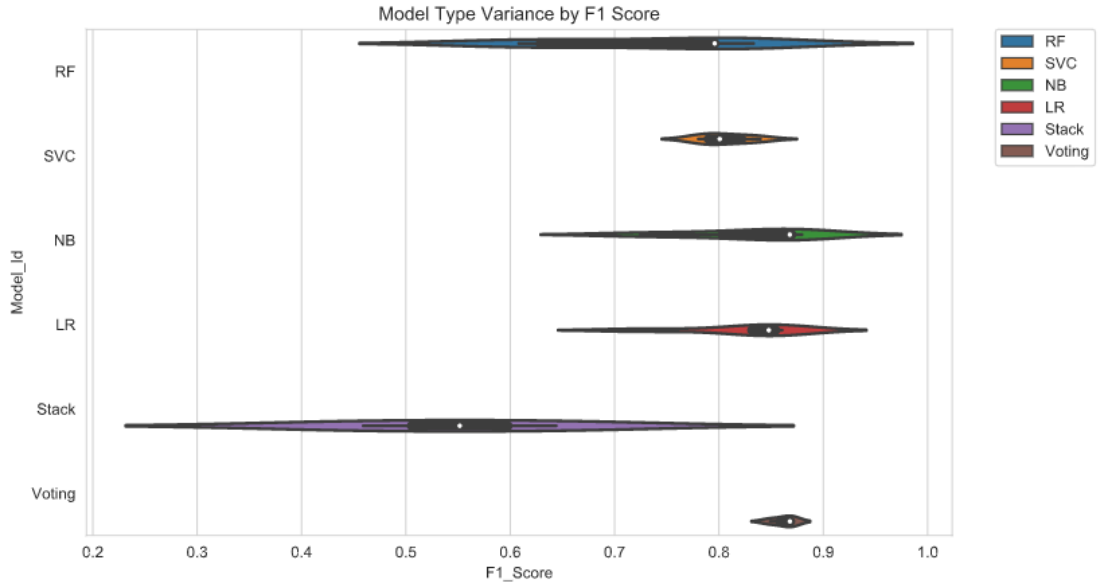
Comparison of the Progression from baseline thru ensemble learning

Comparison and Analysis

- Confusion Matrix from M1 and M2
- **Comparison of the progression: from baseline thru ensemble**
- Comparison of 2000 & 4000 records: Learning Curves and F1-Score (covers the bias-variance tradeoff)

Interpreting and Explaining Models

- Feature Importance
- Individual Tree Visualization
- ELI5
- LIME
- Skater



Comparison of the Progression from baseline thru ensemble learning

The RF, SVC, NB and LR violins are showing the variance from the baseline thru the ensemble learning for the different model types. The general trend is an improvement up to the tuning of the hyperparameters.

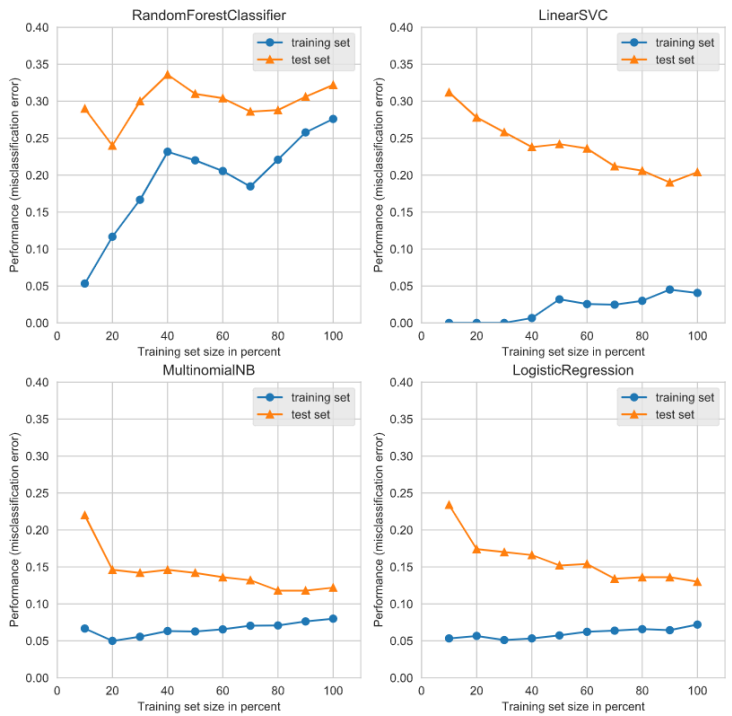
Generally, the ensemble learning methods did not score better than the models with optimized features and hyperparameters tuned. However the voting ensemble almost matched the best tuned models.

Comparison and Analysis

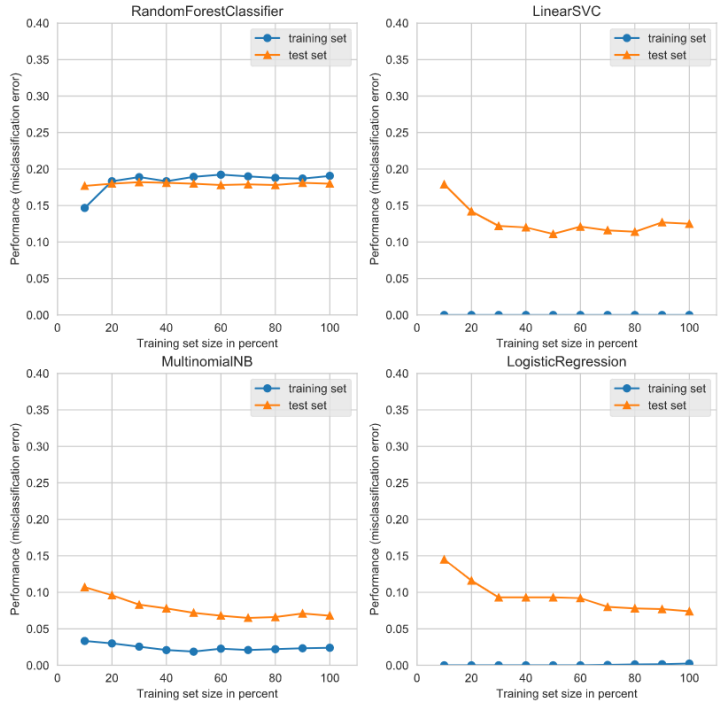
- Confusion Matrix from M1 and M2
- Comparison of the progression: from baseline thru ensemble
- **Comparison of 2000 & 4000 records: Learning Curves and F1-Score (covers the bias-variance tradeoff)**

Interpreting and Explaining Models

- Feature Importance
- Individual Tree Visualization
- ELI5
- LIME
- Skater



Training /Testing Errors 2000 Samples



Training /Testing Errors 4000 Samples

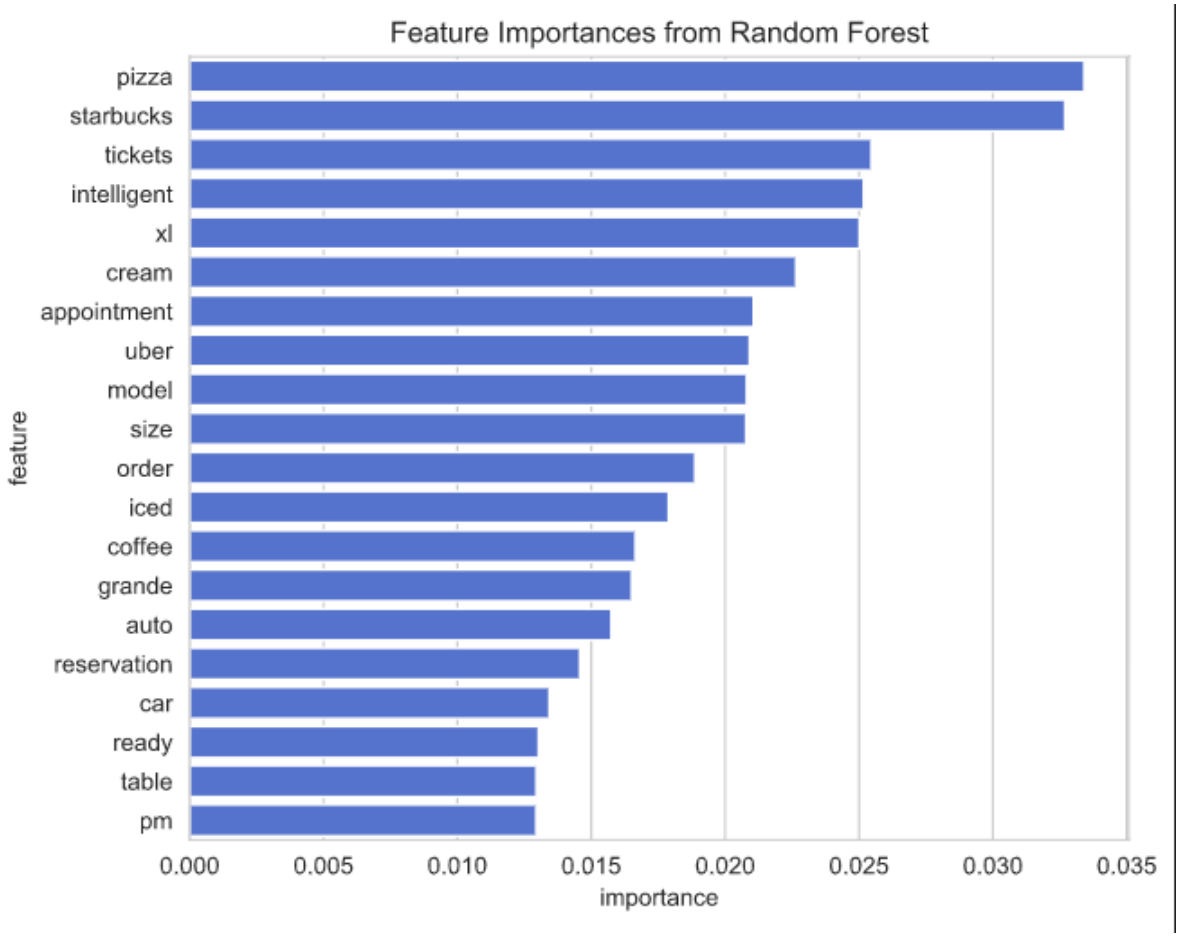
- Increasing the number of samples reduces the overall error, which reduces bias.
  - The variance gap reduced for all model types
  - The error and bias is reduced for all model types

Comparison and Analysis

- Confusion Matrix from M1 and M2
- Comparison of the progression: from baseline thru ensemble
- Comparison of 2000 & 4000 records: Learning Curves and F1-Score (covers the bias-variance tradeoff)

Interpreting and Explaining Models

- **Feature Importance**
- Individual Tree Visualization
- ELI5
- LIME
- Skater



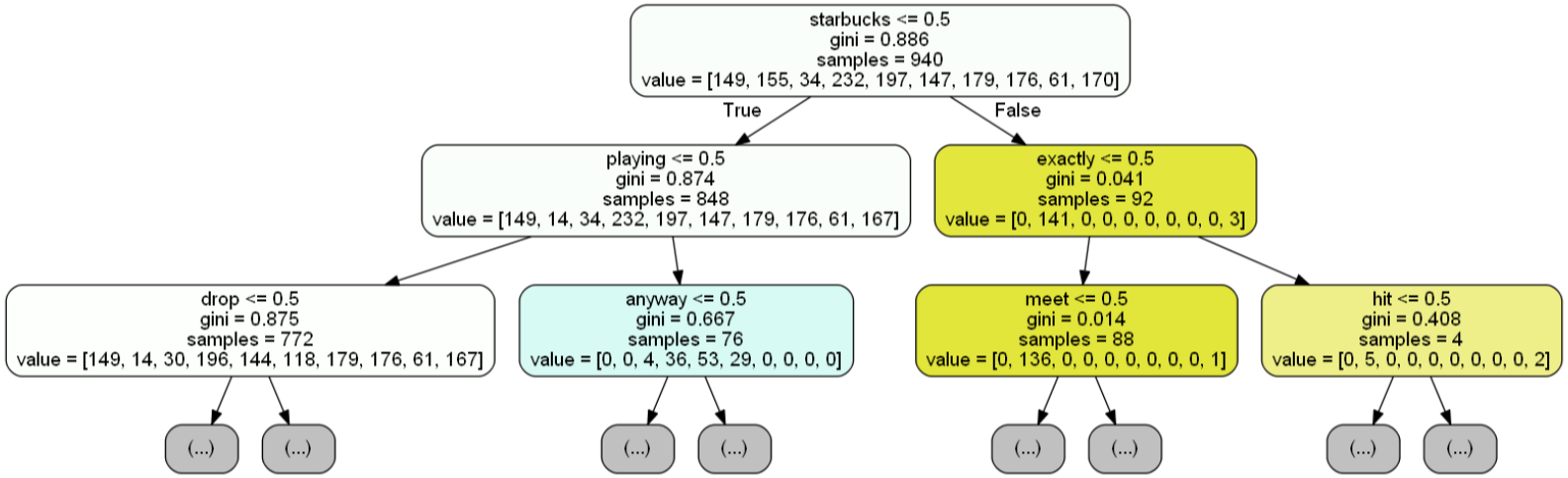
- **Feature Importance**: lists the importance factor of specific words in the corpus in determining the classification
  - Words identified show correlation with the classes

Comparison and Analysis

- Confusion Matrix from M1 and M2
- Comparison of the progression: from baseline thru ensemble
- Comparison of 2000 & 4000 records: Learning Curves and F1-Score (covers the bias-variance tradeoff)

Interpreting and Explaining Models

- Feature Importance
- **Individual Tree Visualization**
- ELI5
- LIME
- Skater



- **Individual Tree Visualization**: gives insight into the decision path for making a specific prediction

Comparison and Analysis

- Confusion Matrix from M1 and M2
- Comparison of the progression: from baseline thru ensemble
- Comparison of 2000 & 4000 records: Learning Curves and F1-Score (covers the bias-variance tradeoff)

Interpreting and Explaining Models

- Feature Importance
- Individual Tree Visualization
- **ELI5**
- LIME
- Skater

Estimator: Linear SVC

y=auto-repair-appt-1 top features		y=coffee-ordering top features		y=movie-finder top features		y=movie-tickets-1 top features		y=movie-tickets-2 top features	
Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature
+2.060	appointment	+2.207	starbucks	+1.363	seen	+2.049	tickets	+2.161	us
+1.415	car	+1.653	milk	+1.278	comedy	+1.736	glass	+1.644	wonder
+1.366	auto	+1.392	coffee	+1.247	movies	+1.370	theatre	+1.443	captain
+1.076	intelligent	+1.331	latte	+1.184	action	+1.171	popcorn	+1.432	enjoy
+1.013	solutions	+1.072	caramel	+0.980	movie	+1.092	purchase	+1.275	tickets
+0.891	tomorrow	+1.031	venti	+0.873	master	*… 1649 more positive …*		+1.204	sent
+0.843	number	+1.025	cream	+0.770	something	*… 3355 more negative …*		+1.199	marvel
+0.780	tune	+0.890	drink	+0.679	watch	-1.449	sorry	+1.095	oclock
+0.746	tires	+0.846	size	*… 955 more positive …*		-1.473	sold	*… 1568 more positive …*	
… 1309 more positive …		*… 1011 more positive …*		*… 3354 more negative …*		-1.593	shazam	*… 3044 more negative …*	
… 3481 more negative …		*… 3661 more negative …*		-0.682	tickets	-1.783	dumbo	-1.363	buy
-0.777	<BIAS>	-1.036	pizza	-0.755	pm	-2.074	us	-1.643	glass

y=movie-tickets-3 top features		y=pizza-ordering top features		y=restaurant-table top features		y=restaurant-table-3 top features		y=uber-lyft top features	
Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature
+1.973	sorry	+2.525	pizza	+2.110	reservation	+1.191	try	+2.768	uber
+1.797	pet	+1.175	pizzas	+1.256	booth	+1.115	cancel	+2.128	ride
+1.743	shazam	+1.127	large	+1.245	italian	+1.026	date	+1.833	lyft
+1.585	hellboy	+1.035	cheese	+1.206	table	+1.024	another	+1.454	uberx
+1.369	movie	+1.019	crust	+0.987	sunday	+1.013	sorry	+1.193	airport
+1.202	buy	+0.966	pepperoni	*… 1829 more positive …*		+0.945	bjs	+1.183	xl
+1.182	cancel	+0.897	order	*… 3282 more negative …*		*… 1029 more positive …*		+1.139	lux
+1.169	dont	+0.889	peppers	-0.989	cancel	*… 2974 more negative …*		+0.828	trip
… 1095 more positive …		+0.787	sausage	-1.067	movie	-0.962	two	+0.783	fare
… 2938 more negative …		*… 992 more positive …*		-1.141	sorry	-0.975	movie	*… 1307 more positive …*	
-1.159	text	*… 3661 more negative …*		-1.352	another	-0.995	italian	*… 3596 more negative …*	
-1.416	sent	-0.760	<BIAS>	-1.539	tickets	-1.059	tickets	-0.880	tickets

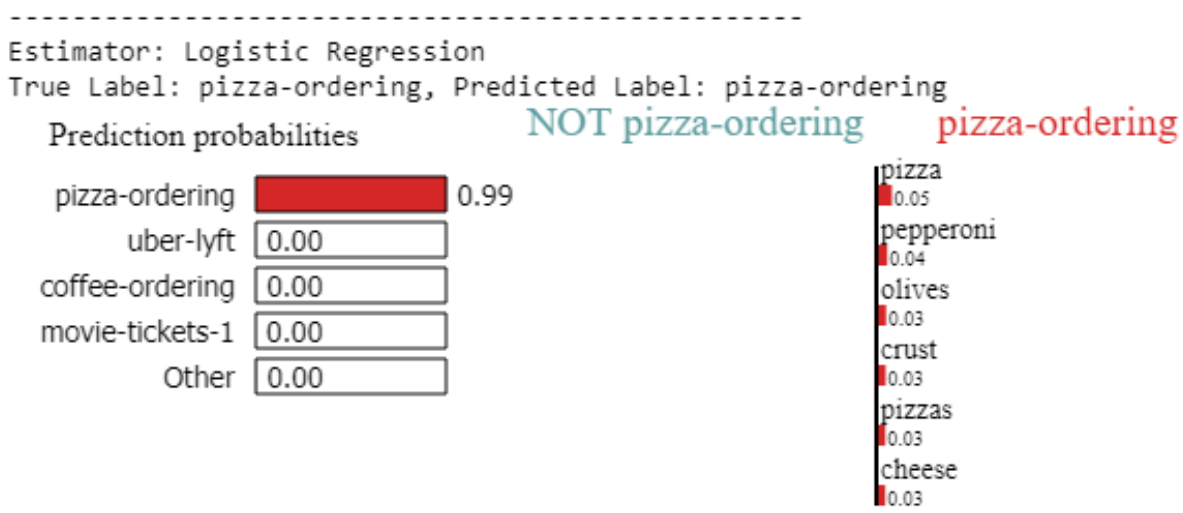
- **ELI5**: Shows the feature importance weights (both negative and positive) for the different classes

Comparison and Analysis

- Confusion Matrix from M1 and M2
- Comparison of the progression: from baseline thru ensemble
- Comparison of 2000 & 4000 records: Learning Curves and F1-Score (covers the bias-variance tradeoff)

Interpreting and Explaining Models

- Feature Importance
- Individual Tree Visualization
- ELI5
- **LIME**
- Skater



Text with highlighted words

need order five pizzas pizza hut pizza hut one closest home okay sizes want five large types crust three pan two handtossed toppings want first pan pizza pepperoni salami ham sausage bacon second one black olives bell pepper mushrooms third salami ham black olives tomatoes bell pepper got want first hand tossed pizza pepperoni black olives mushrooms want second second half half whats first half bell pepper tomatoes sausage whats second half pepperoni chicken beef pork okay got want extra cheese want normal amounts cheese special crusts dont want anything special crust okay got need drinks sides need pizzas everything yes thank thank ill place order right away

- **LIME**: Visualize the words used in the document as well as their associated weights for making prediction on a local document

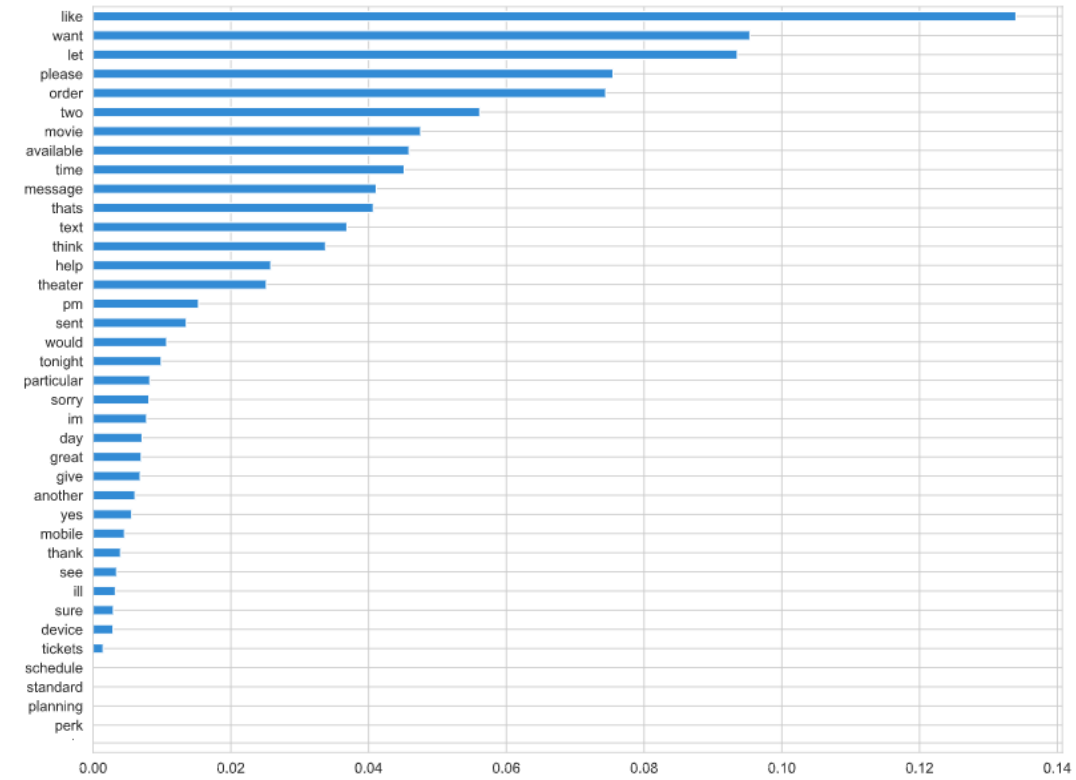
Comparison and Analysis

- Confusion Matrix from M1 and M2
- Comparison of the progression: from baseline thru ensemble
- Comparison of 2000 & 4000 records: Learning Curves and F1-Score (covers the bias-variance tradeoff)

Interpreting and Explaining Models

- Feature Importance
- Individual Tree Visualization
- ELI5
- LIME
- **Skater**

```
---------------------------------------------------
Estimator: Linear SVC Prob
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
[74/74] features ████████████████████ Time elapsed: 1 seconds
```



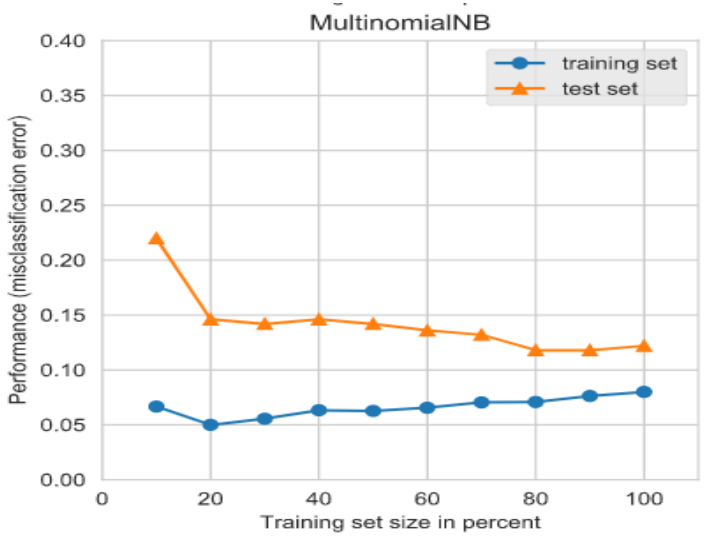
- **Skater**: lists the importance factor of specific words in the corpus in determining the classification
  - Words identified show correlation with the classes

Recommendations

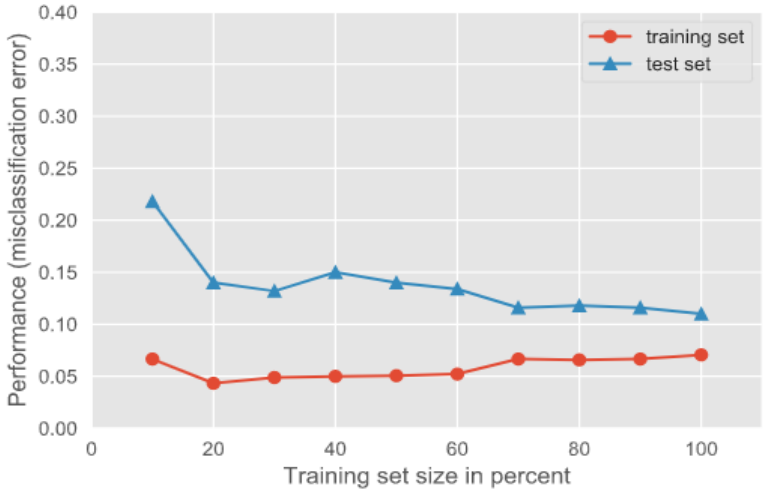
- **Discussion of Naïve Bayes with Hyperparameters and Voting**
- Model chosen

Future Work

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
10	NB	MultinomialNB	optimized	tuned	{'alpha': 0.44, 'fit_prior': False}	0.8794765
24	Voting	Weighted Voting soft	optimized	tuned		0.8692584



Training /Testing Errors Multinomial NB



Training /Testing Errors Weighted Voting

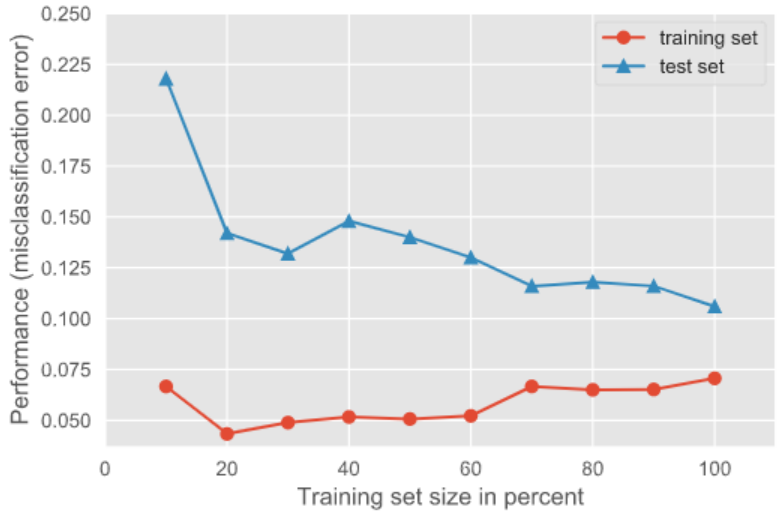
- The Naïve Bayes with optimized features and tuned hyperparameters and the ensemble weighted voting soft gave us our best results.
- Both the variance and bias are slightly improved for the ensemble voting model, indicating a more robust model.

Recommendations

- Discussion of Naïve Bayes with Hyperparameters and Voting
- **Model chosen**

Future Work

	Model_Id	Model	Features	Hyper_Param	Best_Params	F1_Score
10	NB	MultinomialNB	optimized	tuned	{'alpha': 0.44, 'fit_prior': False}	0.8794765
24	Voting	Weighted Voting soft	optimized	tuned		0.8692584



- Based on our results, the recommended model to proceed with is:
  - **<u>Weighted Voting Ensemble (Soft Voting)</u>**

Recommendations

- Discussion of Naïve Bayes with Hyperparameters and Voting
- Model chosen

Future Work

- Looking at other feature selection methods
- Look at adding language models:
  - BERT
  - ELMO
- Newer model types:
  - RNN
  - LSTM
  - Deep Learning
- Deployment of the model

References / Bibliography

Upasana, Mar. 17, 2017, Imbalanced Data : How to handle Imbalanced Classification Problems
https://www.analyticsvidhya.com/blog/2017/03/imbalanced-data-classification/

Li, Susan, Feb 19, 2018, Multi-Class Text Classification with Scikit-Learn
https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f

Sarkar, Dipanjan (DJ), Jan 6, 2018, Categorical Data, Strategies for working with discrete, categorical data
https://towardsdatascience.com/understanding-feature-engineering-part-2-categorical-data-f54324193e63

Jaadi, Zakaria, September 4, 2019, Updated: February 24, 2020, A Step By Step Explanation Of Principal Component Analysis
https://builtin.com/data-science/step-step-explanation-principal-component-analysis

Powell, Victor, Lehe, Lewis, Principal Component Analysis, Explained Visually
https://setosa.io/ev/principal-component-analysis/

scikit-learn 0.22.2, Receiver Operating Characteristic (ROC)
https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

Vallantin, Lima, Oct 9, 2018, Metrics to measure machine learning model performance
https://medium.com/@limavallantin/metrics-to-measure-machine-learning-model-performance-e8c963665476

Borges, Diogo Menezes Jan. 2019, Ensemble Learning: 5 Main Approaches
https://www.kdnuggets.com/2019/01/ensemble-learning-5-main-approaches.html

Smolyakov, Vadim, Aug. ,22, 2017, Ensemble Learning to Improve Machine Learning Results,
https://blog.statsbot.co/ensemble-learning-d1dcd548e936

Koehrsen, Will, May 18, 2018, A Complete Machine Learning Walk-Through in Python: Part Three, Interpreting a machine learning model and presenting results
https://towardsdatascience.com/a-complete-machine-learning-walk-through-in-python-part-three-388834e8804b

Hulstaert, Lars, Feb 20, 2018, Interpreting machine learning models
https://towardsdatascience.com/interpretability-in-machine-learning-70c30694a05f

References / Bibliography

Ma, Edward, Jul 29, 2018, 3 ways to interpretate your NLP model to management and customer
https://towardsdatascience.com/3-ways-to-interpretate-your-nlp-model-to-management-and-customer-5428bc07ce15

Li, Susan, Jul 3, 2019, Explain NLP models with LIME & SHAP, Interpretation for Text Classification
https://towardsdatascience.com/explain-nlp-models-with-lime-shap-5c5a9f84d59b

scikit-learn 0.23.0, 3.2. Tuning the hyper-parameters of an estimator
https://scikit-learn.org/stable/modules/grid_search.html#grid-search

NLP in Python - Quickstart Guide
https://github.com/NirantK/nlp-python-deep-learning

Kessler, Jason, Beautiful visualizations of how language differs among document types.
https://github.com/JasonKessler/scattertext

Seaborn, 0.10.1, Building structured multi-plot grids
https://seaborn.pydata.org/tutorial/axis_grids.html

AlammarThe, Jay, Dec. 3, 2018, Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)
https://jalammar.github.io/illustrated-bert/

Brownlee, Jason, May 20, 2016, Feature Selection For Machine Learning in Python
https://machinelearningmastery.com/feature-selection-machine-learning-python/

Brownlee, Jason, July 14, 2014, Feature Selection in Python with Scikit-Learn
https://machinelearningmastery.com/feature-selection-in-python-with-scikit-learn/

BY DATAFLAIR TEAM · UPDATED · OCTOBER 4, 2019
https://data-flair.training/blogs/machine-learning-classification-algorithms/

https://machinelearningmastery.com/how-to-reduce-model-variance/

Eryk Lewinson, Ensemble learning using the Voting Classifier
https://levelup.gitconnected.com/ensemble-learning-using-the-voting-classifier-a28d450be64d