Predicting SLAs from Annotator Time Per Task with Multiple Linear Regression

Mariam Joan
Loyola Marymount University, Los Angeles, CA USA
mjoan@lion.lmu.edu

Abstract

Data labeling companies have become widely used in the tech sector given rise of big data, machine learning, and a focus on data quality. These companies can be used to provide labels or annotations to improve data quality and or provide training data to machine learning teams. Given the high cost of labeling, with many AI initiatives such as with computer vision, relying on millions of labels for training, most companies move toward a crowd sourced solution such as with AWS Mechanical Turk which can start anywhere from 0.01 *per task and up. However, there are companies who are willing to invest more in exchange for a dedicated team of annotators to work solely on their specific data use cases. In this case, these labeling teams build intuition over time with given tasks, and thus can become more reliable, and efficient. The goal of this research is to predict service level agreements (SLAs) of labeling jobs from such labeling teams, to be able to provide partners and stakeholders streamlined project timelines. Teams relying on this data want to know when labeling jobs will be completed as to begin their model training and or pull the labeled data from a data warehouse, that is now of higher accuracy.*

1 Introduction

The data used for this experiment and model training was provided by an external labeling company with email IDs obfuscated. The data consisted of two csv files totaling over 50k rows, with feature names (see Table 1) representing data pertaining to each project which included minutes and seconds per task. There were a number of considerations prior to deciding on a model algorithm which were how to incorporate time series into model training given this data has time series properties, such as start task and end task timestamps.

When researching how to model time series data, there are many statistical packages such as `https://www.statsmodels.org` in addition to mathematical formulas of which to apply to time series data such as Autoregressive Integrated Moving Average (ARIMA), and Autoregression (AR), see more in (see Table 2). Another interesting aspect of time series is the seasonality and curve to the data such that another tactic one can use is applying the sin and cosine formulas to your model weights, given principles from Fourier transform to show cyclical nature of our time series.(1) However, because the data used in this project is constrained to less than six months we will lack the benefit of seeing any type of annual or seasonal pattern although we will parse the day of the week and could view patterns at a weekly granularity.

While researching this topic of predicting annotator times, it was found that the majority of current research is devoted to crowd sourced solutions while little to no results were found on working with full time employee annotators in any capacity. It's interesting that was the case given companies like `https://www.sama.com/`, `https://www.appen.com/`, and `https://www.labelbox.com/` provide full time employee dedicated annotation teams at a tiered cost. On a positive note, the impetus behind `https://www.sama.com/` founded by `https://www.leilajanah.com/` is that even third world countries where it's difficult to find work, and because annotation tasks tend to be lower complexity, this work can be outsourced where the gig economy in these countries, can take advantage. This offers the opportunity of an annotator to get paid a decent wage for their work vs. 0.01 cost per task as a Mechanical Turk.

Coming back to topics related to crowd sourced annotations, typically of which include quality of annotations as it relates to type of task, cost per task, and using mathematical algorithms such as Expectation Maximization to determine if a model could outperform the annotator decision, essentially in an effort to further reduce annotation costs, and have the model complete remaining or even all training annotations. Expanding upon that as such

Features	Description
project name	type of project
project total rows	rows per project
total worker labels	labels annotators completed
worker performance	skill set range scale 1-10
complexity	task complexity 1-10, highest difficulty=10
worker id	email identifier
worker role	reviewer or annotator
minutes per task	total minutes to complete (1) task
accepted labels	project accepted annotations per worker
rejected labels	project rejected annotations per worker
am pm	time of day task completed
date start day	day of week task completed
date start month	month task completed

Table 1: Feature guide.

Table 2 Statistical Methods for Time Series
Autoregression (AR)
Moving Average (MA)
Autoregressive Moving Average (ARMA)
Autoregressive Integrated Moving Average (ARIMA)
Seasonal Autoregressive Integrated Moving-Average (SARIMA)
Vector Autoregression (VAR)
Vector Autoregression Moving-Average (VARMA)
Vector Autoregression Moving-Average with Exogenous Regressors (VARMAX)
Simple Exponential Smoothing (SES)
Holt Winter's Exponential Smoothing (HWES)

Table 2: Time Series Model Methods[1]

with Dawid and Skene's research which looked at the maximum likelihood of a correct classification based on previous classification, a recent Amazon Web Services article discussed the addition of worker performance as a predictor of label quality. This ultimately was the inspiration to include worker performance in this experiment.(2)

2 Data Pre-Processing

Transforming the data occurs during pre-processing, and for this project we are utilizing Python programming language, which has helpful libraries such as `https://numpy.org/` and `https://pandas.pydata.org/`. In this case, we converted data with timestamp properties into datetime objects and created additional columns from these timestamps such as day of week, day part (AM/PM), minute per task, and second per task. This allowed us to create our target variable which is seconds per task. It's possible that these particular aspects of the labeling task environment (time of day, day of week) could affect the total seconds it takes to complete a task.

2.1 Exploratory Data Analysis (EDA)

After pre-processing it's good practice to visualize our data, especially to check for linear relations between our dependent and independent variables. This process is called, exploratory data analysis or EDA. To begin, in a pair plot, the most worker performance scale is between 8 and 9 range out of a scale of 1 to 10 with 10 being highest performance. While for histogram feature plot we see more skewed distributions (see Figure 1) and also we mainly had higher complex tasks than lower ones. We also see these tasks were completed mostly equal between AM and PM hours, with majority of tasks completed close to under 5 minutes, and the worker role was about even which were either annotator or reviewer.

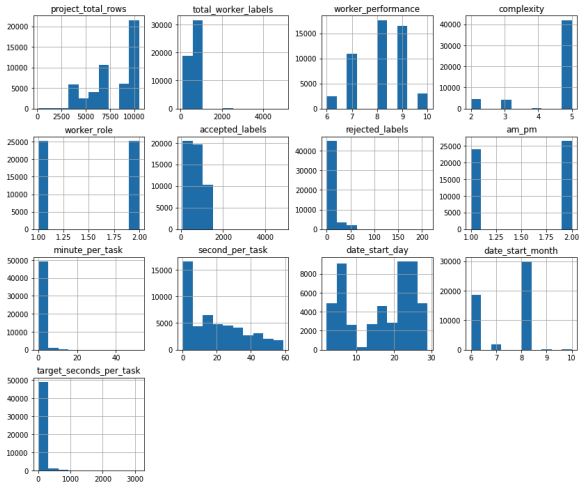

Figure 1: Histograms

Additionally, for the correlation plot (see Figure 2) we do see higher correlation between project total rows, accepted labels, and total worker labels, along with minute per task. Given the complexity is more important than the job name, we can drop the job descriptions as complexity is what we want to learn from, e.g. that it impacts seconds per task. If a task is less complex our hypothesis is that the job would be done much faster than if it were more complex.

2.2 Categorical Encoding

At this point we have performed a train test split on our data, which in this experiment was an 80/20 split. Now, we will look to encode any categorical data which converts it to a numeric form. There are typically three types of categorical features, which are either ranked or in order such as salary, educational credentials, then numerical values or strings that have no relation between each other such as with countries, zip codes, and finally a binary output of either on, off or true, false.

As we consider which columns require label encoding, we know that we want the model to learn each annotator's performance so the worker id or email identifier will be important. The project name although interesting and related to the task,

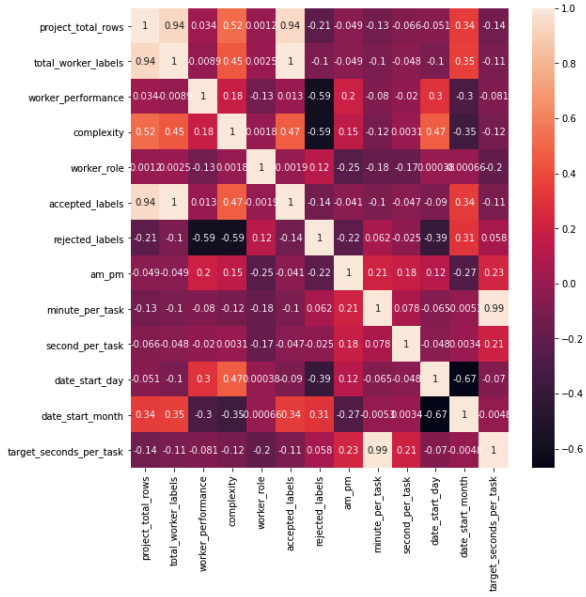

Figure 2: Correlation Matrix

it's ultimately the task complexity that is most important so we can drop the project name feature.

We will use `https://scikit-learn.org/` One Hot Encoder to encode worker id and then concatenate the transformed data with our original data replacing worker id with the new encoded worker id column.

3 Model Constraints and Assumptions

Deciding on which baseline model algorithm to move forward with was fairly simple given our target variable is continuous and that lends itself well to a regression model, while adding in the linearity of our data, and multiple independent variables. We will go with a manual implementation of Multiple Linear Regression to start our model process. The algorithm used was based on an model class implementation by Dario Radečić and edited for various purposes(3).

There are assumptions to consider when working with a multiple linear regression model in order to ensure best outcome with model interpretability. These assumptions are listed below.

- Tending toward or mostly normally distributed.
- Independent variables are not linearly related themselves.
- Minimal to no outliers.
- No multicollinearity.

From the histogram, there were few skewed data distributions, that were in essence bypassed at first, focusing only on outliers for project total rows, total worker labels, and accepted labels. It was decided to move forward despite these findings. However, the correlation plot we saw before, did show highly correlated variables, which proved our data does in fact, exhibit multicollinearity. What's next, how do we resolve this, and where to go from here?

3.1 Multicollinearity

While researching this topic, articles and academic abstracts, even from as far as 25 years ago, we see authors state very similar procedures for detection although newer yet also similar techniques for resolution of multicollinearity, which is when the independent variables themselves are correlated, so interpreting the affect of the variable to the target variable becomes obscured. An example, might be predicting salary, where coefficients b1, and b2 are years of experience and age, respectively. It might difficult to determine which is impacting salary, was it age or years of experience?

Detection of multicollinearity can start with a correlation plot, there are other techniques as well such as variance inflation factor (VIF). This mathematical formula substitutes the target variable with each coefficient, finding the R2 value, and in your result anything higher than a 5 would be problematic or the variance is inflated.(4) The VIF of the ith variable is

$$VIF_i = \frac{1}{1 - R^2} \tag{1}$$

(5) Applying VIF to our feature set, we saw that the project total rows exhibited the most inflation.

3.2 Principle Component Analysis

There are ways to resolve multicollinearity, of which does include dropping the inflated variable, however because that could cause omitted variable bias(6) it's suggested to instead resolve the multicollinearity by other means. Current research also points to using Ridge Regression or Baysian estimation.(7)

Enter principal component analysis or PCA, which is a dimensionality reduction technique, and can also be utilized for feature extraction. Using the `https://scikit-learn.org/` PCA library we can understand the "..percentage of variance explained as the dimensionality decreases."(8) by

	variance_inflation_factor	column_name
0	1.116537	second_per_task
1	1.158356	minute_per_task
2	1.311488	am_pm
3	2.024306	worker_role
4	2.214127	date_start_day
5	2.513975	worker_performance
6	3.328971	date_start_month
7	3.771415	complexity
8	10.850098	project_total_rows

Figure 3: Variance Inflation Factor

viewing the explained variance ratio which can be viewed in Figure 4. Viewing Figure 5 we can see that as we approach 20 principal components, we are capturing our highest explained variance before then leveling off.

```
array([0.15453588, 0.29257111, 0.38309373, 0.44905424, 0.5049403 ,
       0.55627404, 0.60557242, 0.65329168, 0.70007824, 0.74630371,
       0.7896946 , 0.83200224, 0.86834941, 0.90237469, 0.9298639 ,
       0.94956273, 0.96688869, 0.97978458, 0.99131604, 0.99713438,
       1.        , 1.        , 1.        ])
```

Figure 4: Cumulative Sum of Explained Variance

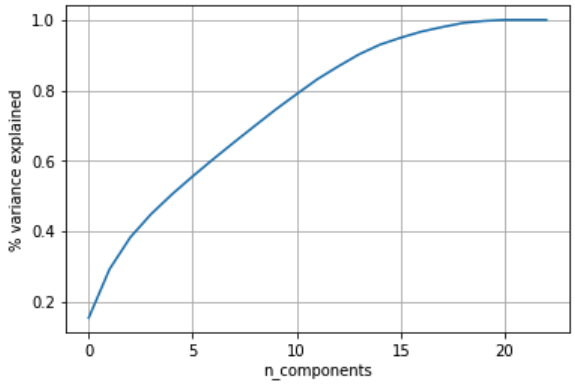

Figure 5: PCA Components Plot

Finally, we can visualize our new optimized dataset that now has no multicollinearity and has also transformed our data into more normally distributed features (Figure 6).

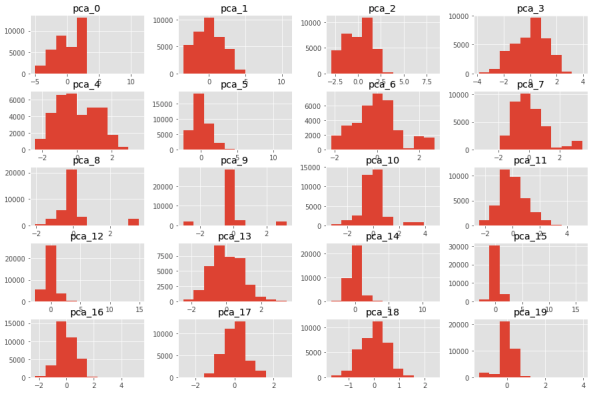

Figure 6: PCA Histograms

4 Modeling

4.1 Multiple Linear Regression

For our baseline model we will implement the algorithm manually where our hypothesis would be:

$$\begin{aligned}\text{TotalSecondsPerTask}_i = \\ &+ \beta_0 \text{Bias}_i \\ &+ \beta_1 \text{ProjectTotalRows}_i \\ &+ \beta_2 \text{WorkerPerformance}_i \\ &+ \beta_3 \text{Complexity}_i \\ &+ \beta_4 \text{DayofWeek}_i \\ &+ \ldots + \beta_n \beta_i\end{aligned}$$

For our cost function we are using Mean Squared Error (MSE) or mathematically as Equation 2.

$$\sum_{\text{i}=1}^{\text{n}} (\text{x}_\text{i} - \text{y}_\text{i})^2 \tag{2}$$

You can see this fully expressed in relation to our thetas in Equation (3).

Our optimizer, gradient descent is used to minimize our cost function and upon each iteration update our theta coefficients which are our weights, and our bias term which is expressed in Equation (4) where brackets reference the equation is repeated until convergence, that is our loss is equal or close to zero or some epsilon value.

$$\text{J}(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \tag{3}$$

$$\left\{ \theta_\text{j} \leftarrow \theta_\text{j} - \alpha \frac{\partial}{\partial \theta_\text{j}} \text{J}(\theta_0, \theta_1) \text{ for j=0,1} \right\} \tag{4}$$

(9)

During running the manual implementation of multiple linear regression, we tested the alpha otherwise known as learning rate, between ranges of 0.5, 0.1 and 0.01 keeping number of iterations constant at 10,000. While the 0.5 proved to be less successful with very large loss calculated by the mean squared error per iteration, the 0.1 and 0.01 did show better results at about the same loss (Figure 7).

Typically we are looking for very low mean squared error, and even root mean squared error (RMSE) or also mean absolute error (MAE). Another consideration was to append the theta coefficients, bias term, and losses per iteration each to their own list as to create a history of each value and plot for interpretation.

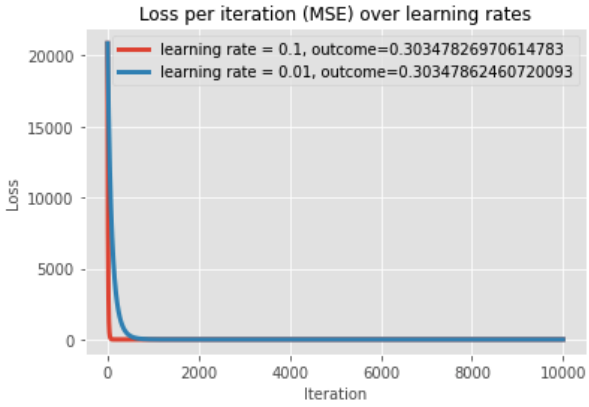

Figure 7: Loss Per Iteration (MSE)

The next model implementations are to compare with our baseline. The following models were implemented using `https://scikit-learn.org/` with exception of (4) which utilized Keras, a Python wrapper API for the Tensorflow library and open source platform.

1. Linear Regression
2. Random Forest Regressor
3. XGBoost Regressor
4. Feed Forward Neural Network

5 Metrics

To evaluate how well our models performed we will use regression metrics such as R^2, MSE, MAE and RMSE as noted in Metrics Tables section, Table 3 and Table 4. The R^2 value, also known as the coefficient of determination, calculates how many observations fall on the regression line, thus if we have a R^2 of 90 percent, then we know that our regression line is fitting the data well. Mathematically, R^2 is calculated as (1 - RSS (Residual Sum of Squares / TSS (Total Sum of Squares)). Additionally, our mean absolute error, or MAE, is calculated as the absolute difference between our actual and predicted values. Similarly, the mean squared error or MSE, is now averaging the squared differences and lastly, root mean squared error is the taking the root of the MSE, this is the average deviation of our actual and predicted values of which we all want these numbers to be on the lower end toward zero. In our metrics tables below we see we have achieved that for our train set however, we are seeing higher metric scores in the test set. This could mean our train data is overfitting (high variance) and test data, underfitting (high bias) although our R^2 is positively stable among both. A possible solution to combat the train overfit, we could reduce our hypothesis feature space, or try adding more data. Given time constraints of this experiment we will leave these as options for a future iteration.

5.1 Metrics Tables

Model	$\alpha = 0.1$	
	R2	**MAE, MSE, RMSE**
MLR (manual)	0.999	0.24, 0.30, 0.55
MLR (sklearn)	0.999	0.24, 0.30, 0.55
RF Regressor	0.999	-, -, -
XGBoost Regressor	0.976	-, -, -
FF Neural Net	0.999	0.44, 0.63, -

Table 3: Train Metrics

Model	$\alpha = 0.1$	
	R2	**MAE, MSE, RMSE**
MLR (manual)	0.996	3.30, 57.13, 7.56
MLR (sklearn)	0.996	3.30, 57.13, 7.56
RF Regressor	0.981	-, -, -
XGBoost Regressor	0.968	-, -, -
FF Neural Net	0.996	3.27, 57.13, -

Table 4: Test Metrics

6 Evaluation

In the final stages of this modeling experiment will focus on methods of evaluation.

6.1 KFold Cross Evaluation

The most common form of evaluation is cross-validation of which there are many types of

including, K-Fold, Stratified, Leave one out (LOOV) and Time Series however, for the purposes of time, we will be using K-Fold from the `https://scikit-learn.org/` library. Here, the train set is split into equal k folds, e.g. divided subsets, and the model is trained using k - 1 folds, while the remaining folds are used as a validation set.(10) Typically the number to pick for k, is either 5 or 10 and shuffling of the data takes place during split as well. The model gets fit on the training data, and evaluated on the validation folds. During each iteration thru the split dataset, it's common practice to keep track of each validation score per iteration and then take the final average of these scores when completed, to give you a baseline performance of the model. In our case, we appended the R^2 values as our model evaluation score. In both train and test K-Fold iterations R^2 performance was 0.999 or less than 0.0001 error.

6.2 Confidence Intervals, p-values

There are other evaluation methods used for multiple linear regression, such as t or F statistic, and p-values. Separately, confidence intervals can also help support model explainability which would be useful in communicating with business partners to validate findings, e.g. within a specified + or - margin of error (MOE) range, our model will predict this coefficient correctly, 95 percent of the time. That will be helpful in this scenario with business partners relying on accuracy of project time durations.

From statistics we know that the p-value, is a probability of obtaining results as extreme as actually observed if null hypothesis is true(11), however, given we are not running hypothesis tests, the p-value can still help us determine statistical significance of our features. Typically, anything less than what we call the alpha or 0.05 would be statistically significant. According to Figure 9, we can see that all of our features are statistically significant, which might be a flag to research further as to why that is. There is a sense that the PCA applied to our dataset, may have been a factor. From Figure 9 we can also view confidence intervals for each feature, although now obscured by the PCA process, we have lost the ability to explain which feature was most valuable to our predictions.

Finally, from Figure 8 we can view the R^2 value which is low at 0.751, interesting how that differs from our previous model metrics. The AIC or The Akaike information criterion is prediction error estimator and can be used for model selection however because we aren't running AIC between other models, this won't be used, however important to know for future evaluations.(12) Furthermore, the BIC or Bayesian information criterion is another model selection technique based on the maximum likelihood estimation method, and similar to the AIC.(13)

OLS Regression Results

Dep. Variable:	y	**R-squared (uncentered):**	0.751
Model:	OLS	**Adj. R-squared (uncentered):**	0.751
Method:	Least Squares	**F-statistic:**	5334.
Date:	Wed, 08 Dec 2021	**Prob (F-statistic):**	0.00
Time:	22:58:11	**Log-Likelihood:**	-2.0157e+05
No. Observations:	35366	**AIC:**	4.032e+05
Df Residuals:	35346	**BIC:**	4.034e+05
Df Model:	20		
Covariance Type:	nonrobust		

Figure 8: R^2, F-statistic, AIC, BIC

7 Conclusions

In conclusion, the next steps for this project involve deployment of the model, further retraining and continued research.

7.1 Retraining

Given the initial multicollinearity experienced early into this project, there are other experiments we could have used, potentially finding other ways to resolve it, possibly through variance inflation factor identification and removal of the suspect features, in addition to using either Ridge Regression or Baysian estimation.

The PCA method was helpful and allowed us to continue with the project, however, one of the aspects that would have been preferred was model interpretability. When using PCA on your dataset, you no longer have access to feature names, as you features essentially turn into principal components which are more abstract, versus the ability to learn about your most important features.

Given non-PCA techniques, we would then run the `https://scikit-learn.org/` package GridSearchCV which is similar to a cross validation that fits and scores the model providing back the best parameters using "best params" method, that gave highest results on hold out data.

In the multiple linear regression model implementation, there are other optimizers that could

	coef	std err	t	P>|t|	[0.025	0.975]
pca_0	-14.4053	0.204	-70.635	0.000	-14.805	-14.006
pca_1	-1.1132	0.216	-5.159	0.000	-1.536	-0.690
pca_2	40.9764	0.266	153.777	0.000	40.454	41.499
pca_3	12.6601	0.312	40.556	0.000	12.048	13.272
pca_4	23.4108	0.339	69.031	0.000	22.746	24.075
pca_5	44.2204	0.354	124.969	0.000	43.527	44.914
pca_6	1.6356	0.361	4.530	0.000	0.928	2.343
pca_7	-12.4324	0.367	-33.875	0.000	-13.152	-11.713
pca_8	-5.1156	0.371	-13.802	0.000	-5.842	-4.389
pca_9	-1.5703	0.373	-4.211	0.000	-2.301	-0.839
pca_10	8.6947	0.385	22.591	0.000	7.940	9.449
pca_11	27.5049	0.390	70.566	0.000	26.741	28.269
pca_12	70.3502	0.421	167.294	0.000	69.526	71.174
pca_13	14.5631	0.435	33.507	0.000	13.711	15.415
pca_14	68.1175	0.484	140.870	0.000	67.170	69.065
pca_15	4.9170	0.571	8.608	0.000	3.797	6.037
pca_16	-3.6205	0.609	-5.944	0.000	-4.814	-2.427
pca_17	4.2543	0.706	6.026	0.000	2.871	5.638
pca_18	-2.8113	0.747	-3.766	0.000	-4.275	-1.348
pca_19	1.3133	1.051	1.250	0.211	-0.747	3.373

Omnibus:	79186.653	Durbin-Watson:	0.000
Prob(Omnibus):	0.000	Jarque-Bera (JB):	992539404.913
Skew:	-20.801	Prob(JB):	0.00
Kurtosis:	822.649	Cond. No.	5.15

Figure 9: Coefficients, Standard Error, t-statistic, p-values Confidence Intervals

have been utilized, such as a stochastic gradient descent, or even incorporate regularization. It also seemed as if 10,000 iterations may have been too many and that the global minimum was able to be reached sooner given the values from the weights and bias iterations and plot to confirm this.

Lastly, it was understood, to learn more about how to apply statistical time series techniques to machine learning algorithms and how it might improve predictions given the features available.

7.2 Deployment

This model will need to be deployed for use, given varying tech stack and software environments, further investigation is required on how to deploy it in such a way that the internal and business teams can enter in specific parameters, such as number of project rows, task complexity and output the total seconds per task.

7.3 Research

More progress needs to be made in the crowd sourced vs full time employee annotator space. There are currently numerous academic papers with references to innovative algorithms that attempt to learn, probabilistically how to label data. This stems from an effort to move away from having to pay for data labeling costs, given projects that require millions of training data rows. Another aspect of not only cost but, also quality. It would be interesting to prove that full time annotators provide best annotated or labeled data quality vs a crowd sourced solution. Would companies be willing to pay extra for better quality completed in less time? The hypothesis is that full time annotators dedicated to your team would build intuition over time as to your recurring project needs, in addition to learning ontologies, or taxonomies.

Acknowledgments

Dr. Mandy Korpusik Thank you for supporting this final project and encouragement during the semester.

Colby Wise, Content Knowledge Graph, Manager Thank you for supporting and allowing use of work time to build the model.

Mary and Kevin McCarthy Thank you for asking how this semester was going and if it was doing well.

References

[1] B. Bateman, A. R. Jha, and I. Mathur, *The Supervised Learning Work-shop, Appendix, Activity 3.04*. Packt Publishing, 2020.

[2] S. Ahmad and L. Moos, “Use the wisdom of crowds with amazon sagemaker groundtruth to annotate data more accurately,” 2019.

[3] D. Radečić, “Master machine learning: Multiple linear regression from scratch with python,” 2021.

[4] P. Allison, “When can you safely ignore multicollinearity?,” 2012.

[5] Wikipedia, “Variance inflation factor,” 2021.

[6] Wikipedia, “Omitted variable bias,” 2021.

[7] C. E. Willis and R. D. Perlack, "Multicollinearity: Effects, symptoms, and remedies," *Journal of the Northeastern Agricultural Economics Council*, vol. 7, no. 1, pp. 55–61, 1978.

[8] S. Kumar, "How to remove multi-collinearity in a dataset using pca," 2020.

[9] R. Jain, "Gradient descent algorithm for linear regression," 2016.

[10] sckit learn, "3.1 cross-validation: Evaluating estimator performance," 2016.

[11] Wikipedia, "P-values," 2021.

[12] Wikipedia, "Akaike information criterion," 2021.

[13] Wikipedia, "Bayesian information criterion," 2021.