Matthew Kligerman, Luis Benitez, Raheema Kadwa

Due December 9, 2018

Machine Learning

Professor Daniel Leeds

How to Get Rich with Machine Learning

Introduction

A subject of particular interest worldwide is the annual income of adults in the workforce. We examined the "Adult Data Set" as provided by the UCI Machine Learning Repository. The data set's earnings column indicates whether each individual makes greater than $50,000 annually or the individual makes less than or equal to $50,000 annually. The data set contains the following continuous attributes: age (years), final weight, number of years of education, capital gain ($), capital loss ($), and hours of work per week. For the final weight, the weights are controlled by independent estimates of the civilian noninstitutional population of the US (prepared monthly by Population Division at the Census Bureau). 3 sets of controls are utilized 1) A single cell estimate of the population 16+ for each state, 2) Controls for Hispanic Origin by age and sex, 3) Controls by Race, age and sex. Furthermore, it also contains the following discrete attributes: work class, education, marital status, occupation, relationship, race, sex, and native country.

Below is the list of possible values of each discrete attribute:

Sex	**Race**	**Relationship**	**Marital Status**	Poland
Female	White	Wife	Married-civ-spouse	Jamaica
Male	Asian-Pac-Islander	Own-child	Married-AF-spouse	Vietnam
	Amer-Indian-Eskimo	Husband	Married-spouse-absent	Mexico
	Other	Not-in-family	Separated	Portugal
	Black	Other-relative	Widowed	Ireland
		Unmarried	Divorced	France
			Never-Married	Dominican Republic
Education	**Occupation**	**Work Class**	**Native Country**	Laos
Doctorate	Tech-support	Private	United States	Ecuador
Masters	Craft-repair	Self-Employed Unincorporated	Cambodia	Taiwan
Bachelors	Other-service	Self-Employed Incorporated	England	Haiti
Some College	Sales	Federal Government	Puerto Rico	Columbia
Associates	Exec-managerial	Local Government	Canada	Hungary
Associates Vocational	Prof-specialty	State Government	Germany	Guatemala
Professional School	Handlers-cleaners	Without Pay	Outlying US (Guam-USVI-etc)	Nicaragua
High School Graduate	Machine-op-inpct	Never Worked	India	Scotland
12th Grade	Adm-clerical		Japan	Thailand
11th Grade	Farming-fishing		Greece	Yugoslavia
10th Grade	Transport-moving		South	El-Salvador
9th Grade	Priv house-serv		China	Trinadad&Tobago
7th-8th Grade	Protective-serv		Cuba	Peru
5th-6th Grade	Armed-Forces		Iran	Hong
1st-4th Grade			Honduras	Holand-Netherlands
Preschool			Philippines	
			Italy	

In undergoing this project, we hypothesized that certain levels of education are requirements for certain jobs that pay more than $50,000 annually, and that ceteris paribus, the higher the level of education (discrete), the higher likelihood of an individual attaining a job paying over $50,000. Education is highly correlated with number of years of education, but they are not the same. For example, people may progress towards degrees at different rates. For example, individuals may be held back and skip grades. Sex, race, relationship, marital status, occupation, and native country are each discrete characteristics related to annual income but do not have a predefined rank of probability in the same manner that education does.

Additionally, the data set lacks multi-valuation of attributes when it is possible for race, relationship, divorced, education, occupation, and work class. People that are multiracial will simply fall into the category of "other" while it may be more accurate to include them in multiple categories. Hispanics, a large ethnic group, are not a possible value for race so they are simply labeled as "other." For relationship, people may have children ("own-child"), while also being married ("husband" or "wife") or being unmarried. It may be valuable to include multiple values for education because individuals may have certain degrees in combination with other degrees. For example, one individual may have gone to a "Professional School" and have a "Bachelors", while another may only have one or the other. Individuals may have multiple occupations and therefore multiple work classes. They may conform to none of the possible values for occupation since the only value for other is "other-service."

Furthermore, this data only includes individuals from certain designated countries from which the data can be derived (listed above). This model is not trained to predict the labels of individuals from other countries. Even when omitting that attribute in making predictions, the training data is skewed the towards the common probabilities of making greater than $50,000 for individuals from those native countries.

In initially reducing features, we determined that final weight and number of years of education are should be removed. Final weight does not improve the results, and the number of years of education is too highly correlated with the education category. We binned age, capital gain, and capital loss, and we calculated the probabilities of making greater than $50,000 for each of the bins.

This data set was published on May 1, 1996, and the correlation between annual income

and the attributes in the data set is time-sensitive. Therefore, these models are best for predicting annual income close to that time period, and likely are unable to replicate accuracy levels quite as high for other time periods. The accuracy of our models in predicting whether an individual makes more or less than $50,000 annually in other time periods, such as in the year 2018, is yet to be tested.

Methods & Process

In order to determine whether an individual would make more or less than $50,000 per year given any possible values of the attributes in the data set, we used a binary classifier. We chose to use the Support Vector Machine (SVM), a supervised learning model which is a non-probabilistic binary linear classifier. The SVM takes the set of training data, marks each to one of the two available labels, >$50k or <=$50k, and creates a clear gap between the two labels which is made to be as wide as possible. The test data is mapped into the same space and the labels are predicted based on which side of the gap they fall on. In order to perform a non-linear classification, we used the Kernel Trick, which implicitly maps inputs into high-dimensional feature spaces. Simply put, shift/transforms the data points so it that the SVM can better separate these values and provide more clearer/accurate predictions.

The parameter C represents the penalty or error term in the SVM equation. In our function, we allowed a parameter to be passed which would classify the model based on either a linear or a quadratic support vector. Typically, we would expect a quadratic support vector to provide more accurate results as it is more fitted to the data. On the other hand, this also poses the risk of overfitting, which may impair the accuracy.

```
def fit_data(data, y, k = 'linear'):
    # Initialization
    num = data.shape[0]

    alpha = np.zeros((num))
    penalty = 1.0
    kernel_type = k

    kernels = {
        'linear' : kernel_linear,
        'quadratic' : kernel_quadratic
    }

    kernel = kernels[kernel_type]
    count = 0
    step = 0.001
    loops =10000

    while True:

        count += 1
        previous_alpha = np.copy(alpha)

        for j in range(0, num):

            i = random(0, num-1, j)
            x_i = data[i,:]
            x_j = data[j,:]
            y_i = y[i]
            y_j = y[j]

            kernel_trick = kernel(x_i, x_i) + kernel(x_j, x_j) - 2 * kernel(x_i, x_j)

            if kernel_trick == 0:
                continue

            hyper_alpha_j = alpha[j]
            hyper_alpha_i = alpha[i]
            (L, H) = get_LH(penalty, hyper_alpha_j, hyper_alpha_i, y_j, y_i)

            #Creating the model
            weights = get_weights(alpha, y, data)
            b = get_b(data, y, weights)

            #obtaining errors
            error_i = error(x_i, y_i, weights, b)
            error_j = error(x_j, y_j, weights, b)

            #obtaining new alpha values
            alpha[j] = hyper_alpha_j + float(y_j * (error_i - error_j))/kernel_trick
            alpha[j] = max(alpha[j], L)
            alpha[j] = min(alpha[j], H)
            alpha[i] = hyper_alpha_i + y_i * y_j * (hyper_alpha_j - alpha[j])

        diff = np.linalg.norm(alpha - previous_alpha)
        if diff < step:
            break
        if count >= loops:
            return
```

```
    b = get_b(data, y, weights)
    if kernel_type == 'linear':
        weights = get_weights(alpha, y, data)

    alpha_loc = np.where(alpha > 0)[0]
    support_vectors = data[alpha_loc, :]

    data_pred = predict(data, weights, b)

    for i, element in enumerate(data_pred):
        if element == -1:
            data_pred[i] = 0

    print('Prediction Values:', data_pred)

    new_y = []
    for i, element in enumerate(y):
        if element == 1:
            new_y.append('>50K')
        else:
            new_y.append('<=50K')


    accuracy = getAccuracy(data_pred, new_y)

    if k == 'linear':
        print('Classify Linear Accuracy:', accuracy)
    else:
        print('Classify Quadratic Accuracy:', accuracy)
    return support_vectors, count
```

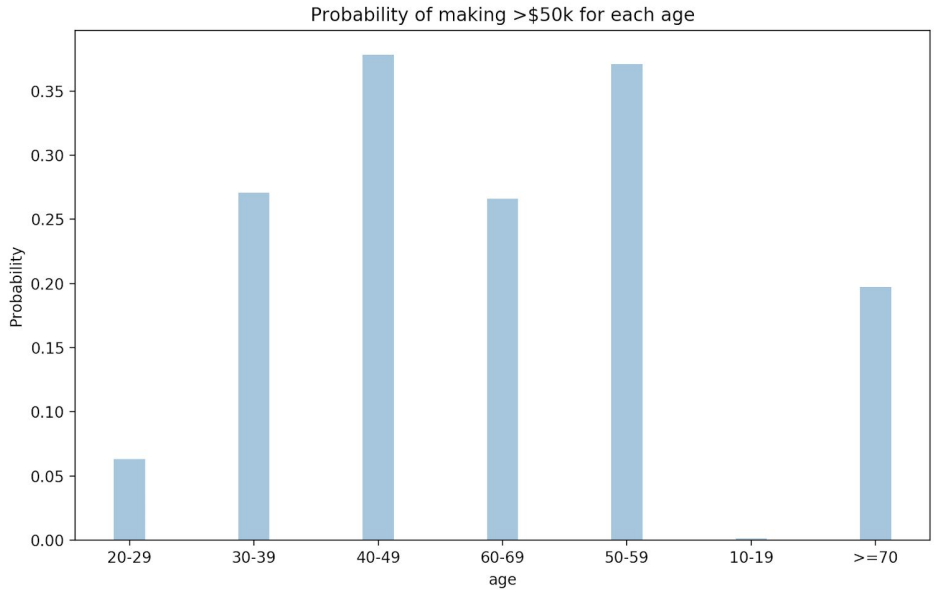
In the code above, we illustrate our fitting function implementation. We first start by determining if we're implementing a linear or quadratic kernel. Next, we start iterating over our data in order use every point to re-adjust the location of our support vector. Note: the maximum number of iterations allowed is 10000 in order to avoid a close-to-infinite loop that will indefinitely reposition the support vector. Then, we proceed to calculate our model parameters (weights and such) error estimates, and alpha values. Lastly, after obtaining our model, we predict the values and calculate the accuracy of the results.

```
svclassifier = SVC(kernel='linear')
svclassifier.fit(list(xtrain.values)[:][:], list(ytrain.values)[:])
ypred = svclassifier.predict(list(xtest.values))

skAccuracy = getAccuracy(ypred[:], list(ytest[:]))
print('Learned Linear Accuracy:', str(skAccuracy))

svclassifier = SVC(kernel='poly', degree = 2)
svclassifier.fit(list(xtrain.values)[:][:], list(ytrain.values)[:])
ypred = svclassifier.predict(list(xtest.values))

skAccuracy = getAccuracy(ypred[:], list(ytest[:]))
print('Learned Quadratic Accuracy:', str(skAccuracy))
```

The above code snippet depicts our Learned Classify code, in which we utilized the sklearn library. The reason behind this decision is to compare these results to those of our own Classification functions. We First start by determining weather we will implement a linear or quadratic (degree to the second power signifies a quadratic equation) expression; in this particular case, we implemented both to see which one will return the most favorable results. Then, We proceed to fit the data points with the sklearn classifier with our train values and the array of correct labels for the earnings column. After the data was properly fitted, we make predictions in both cases of what new possible values could be. Finally, we compare the values predicted to the actual values in our data set, and calculate the accuracy of our model.

Probability results per feature

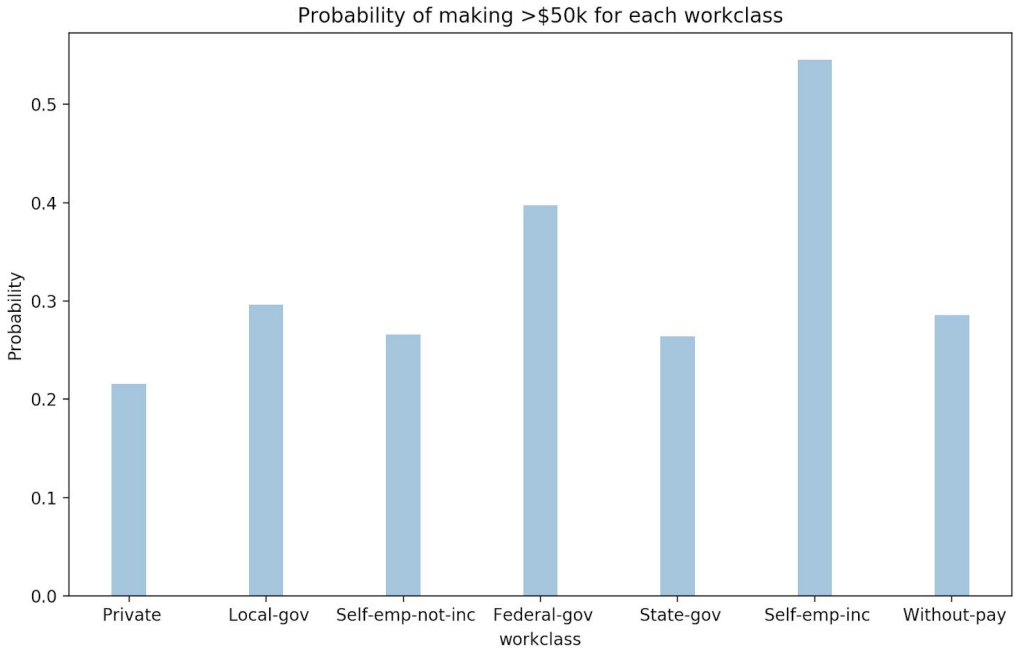
We determined the average probabilities of making more or less than $50,000 per year for each possible value of each discrete attribute (work class, education, marital status, occupation, relationship, race, sex, and native country) so that we could utilize a continuous probability value in lieu of each discrete value. We utilized binning to divide the continuous values into discrete bins, so that we could then calculate the probability values of each bin. The accuracy of this method is superior to calculating the probabilities of each specific continuous value which would provide probabilities that weight each number extremely highly, far more than is optimal. After calculating the probabilities of making greater than $50,000 for each value of each attribute in the training data, we graphed them using the Python's matplotlib library. Below are the graphs that we constructed:



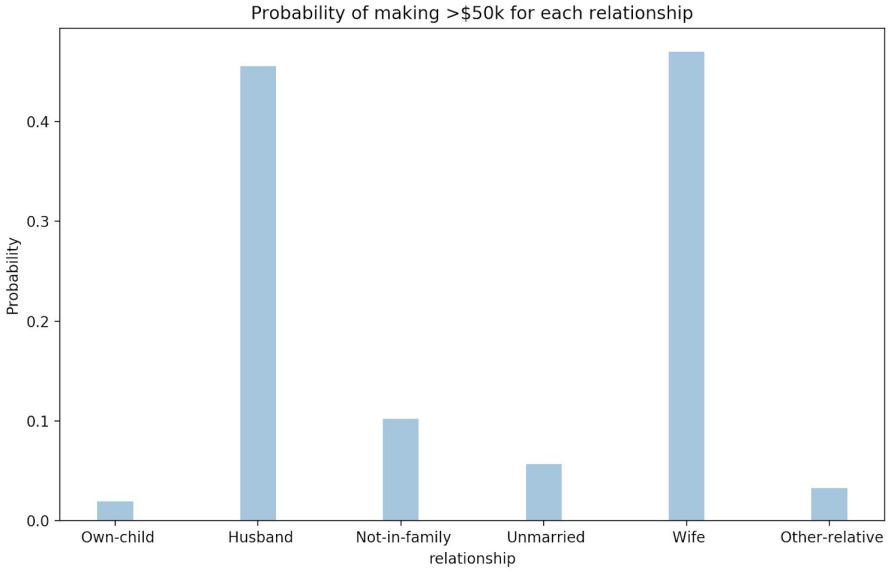
As anticipated, individuals under the age of 20 rarely make more than $50,000 annually, and the probability of earning more than $50,000 increased as individuals got older, peaking at

40-49 years of age, until it began to decrease in every bin afterwards as they individuals grew closer to the age of retirement. In the 1990s, the age of retirement was earlier than it is today, and in 2017, the median age of retirement was 62 years of age.

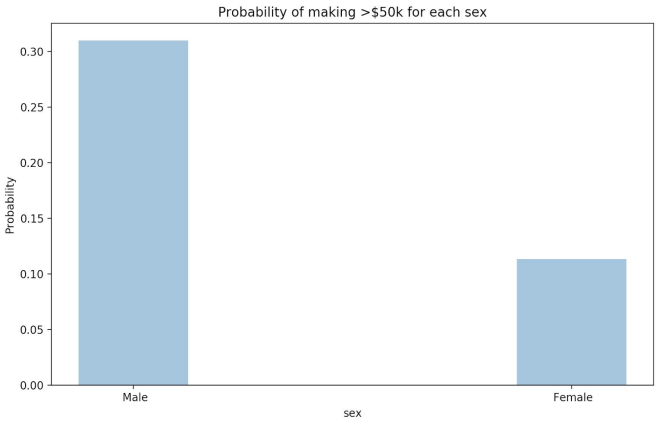
https://dqydj.com/average-retirement-age-in-the-united-states/



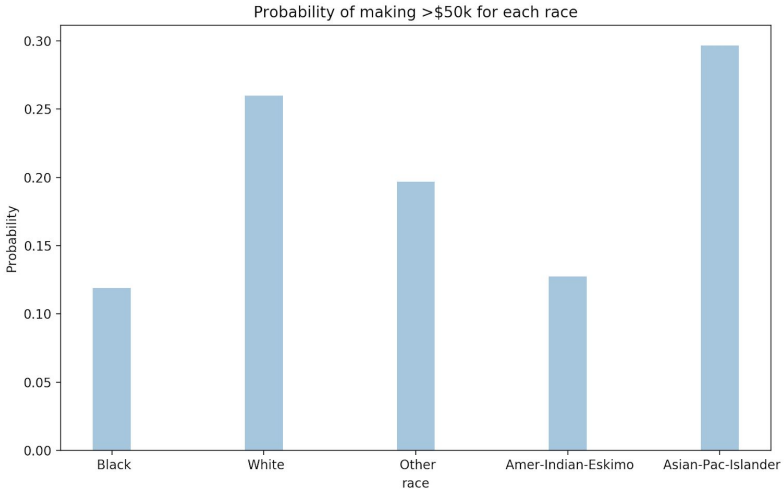
We had no prior expectations of earning probabilities prior to analyzing this data. Nonetheless, we realized that individuals who are self-employed incorporated tend to make more than $50K a year. This result is likely due to the fact they set up their corporation for tax purposes. This set-up provides more credibility for potential clients; thus increasing their stream of revenue.



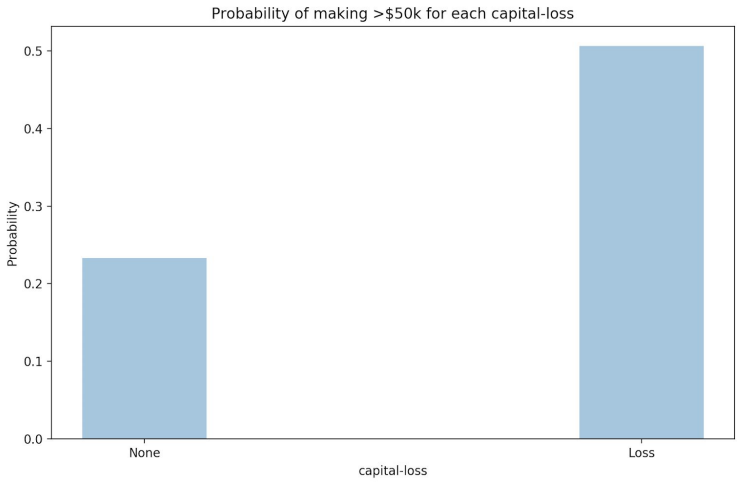
Individuals that were never married, are widowed, separated marital status, divorced, or married spouse absent, all have extremely low probabilities of making more than $50,000. Conversely, married individuals (civilian and armed-forced) had a very high probability of earning more than $50,000. This is highly correlated with the relationship status of the individuals. Although there is definitely between marital status and earnings, we cannot deduce causation. For example, individuals that earn more money may be more likely to get married than individuals that do not, because their partners may marry them partially for financial security.

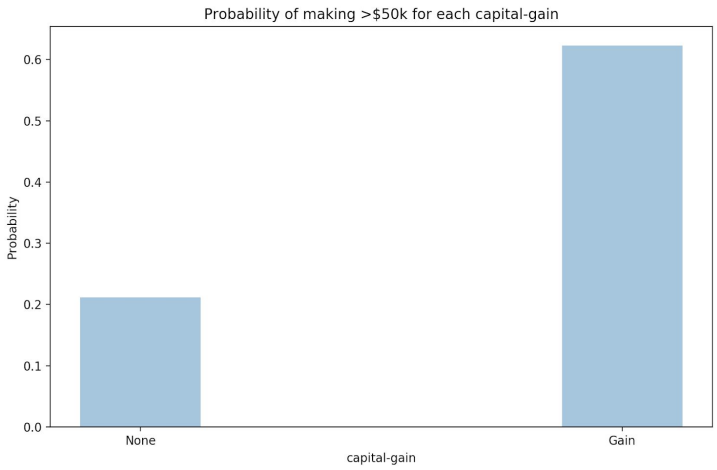


Males had a significantly higher probability of earning over $50,000, with a probability of 30%, as compared to women, with a probability of 10%.

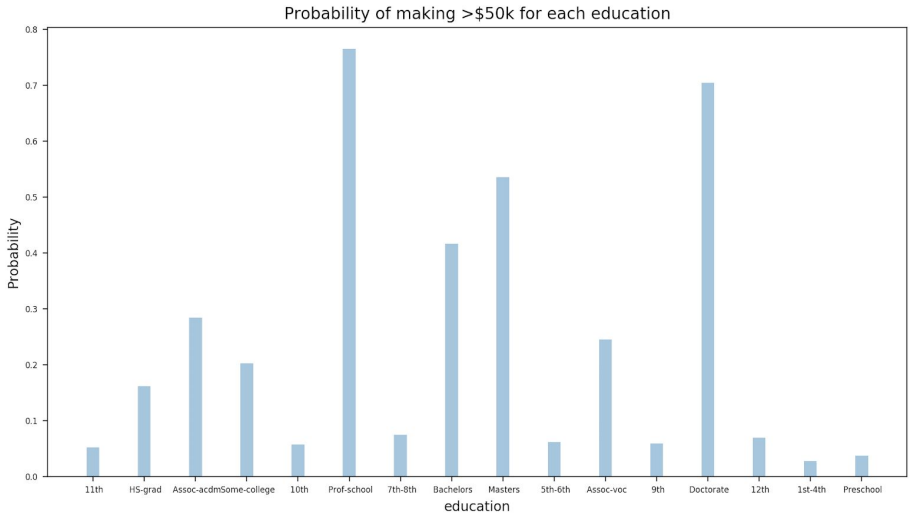


Asian Pacific Islanders attained the higher probability of earning $50,000, followed closely by whites. The order of probability of earning follows as other race, American Indian Eskimo, and black respectively.





We correctly hypothesized that individuals that experienced any capital gain or loss were far more likely to earn more than $50,000 annually, because individuals that have capital gains and losses are typically of higher net worth and thus more likely to have high earnings. We divided the continuous values into two binary bins each, and our results supported our hypothesis, with a great disparity in probability of earning between individuals that had experienced capital gain and/or loss versus those that did not.

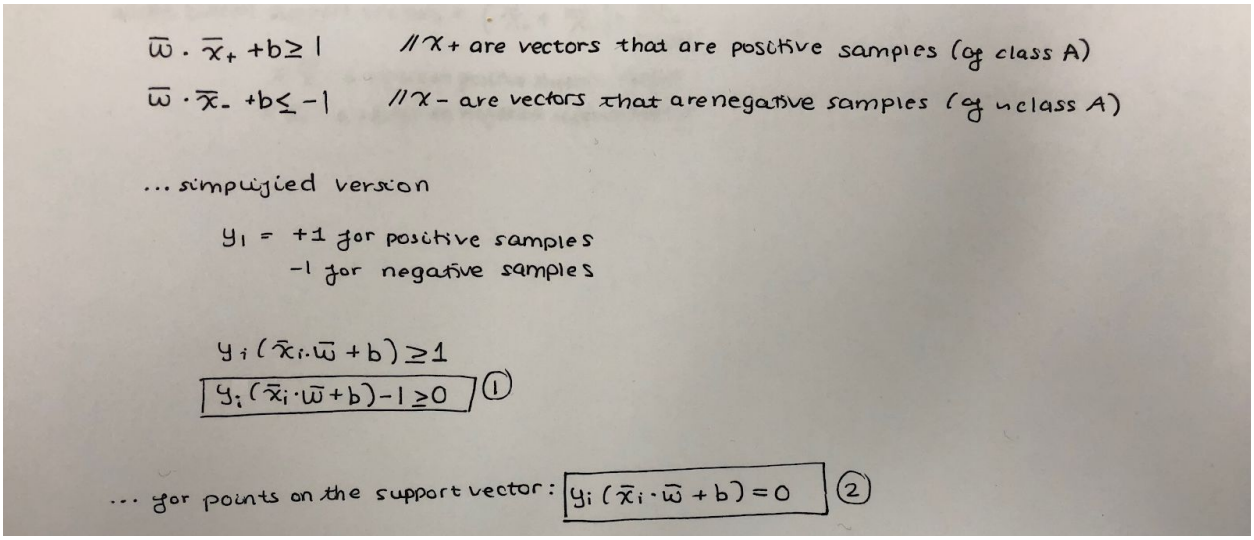


As expected, there was a direct correlation between level of education and probability of earning more than $50,000. These results were almost definite, except for a few outliers, such as those with preschool educations having a higher probability than those with 1st-4th grade educations. Furthermore, professional school graduates were the most likely to earn more than $50,000 because they obtain education specifically targeted at maximizing short-term income growth.

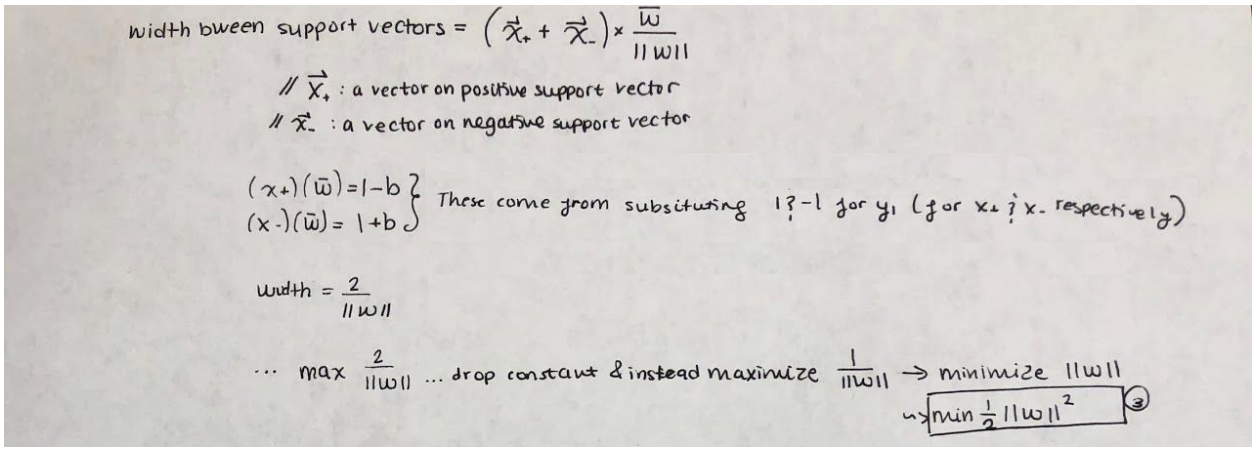
Support Vector Machine Algorithms and Formulas

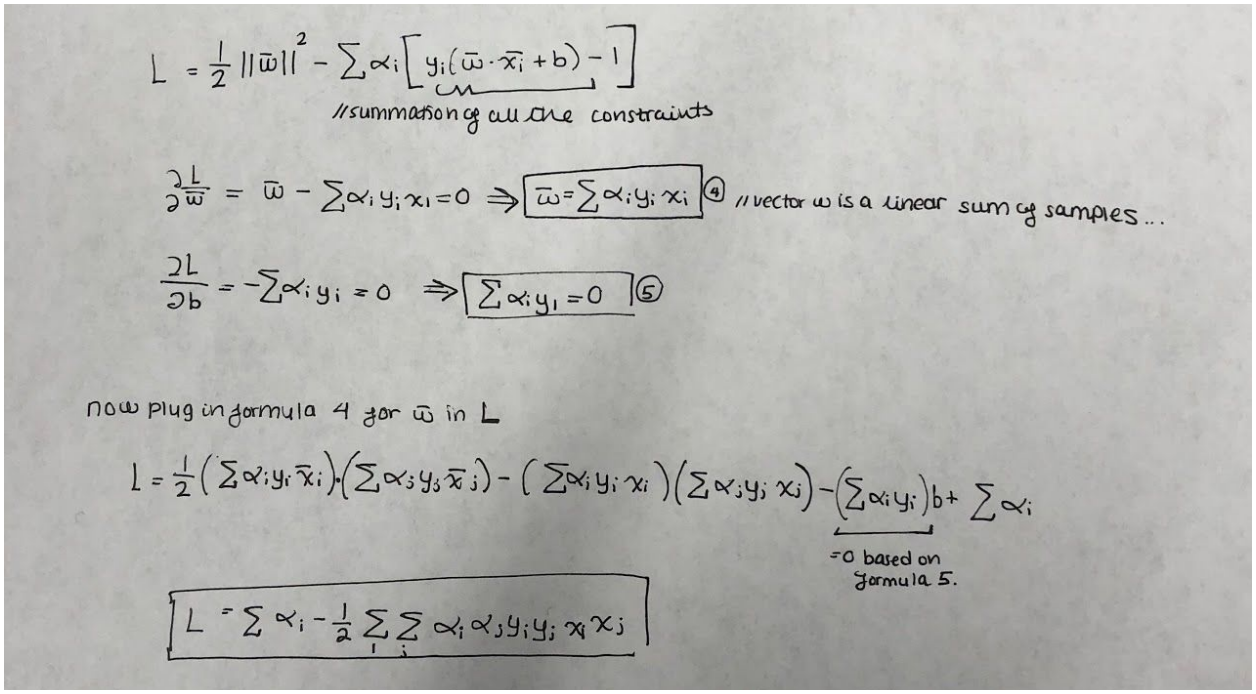
We were able to derive these formulas by following Professor Patrick Winston's lecture on Support Vector Machines on Youtube. The lecture has been cited at the end of our paper.

These are the formulas we learned in class about classifying data points.

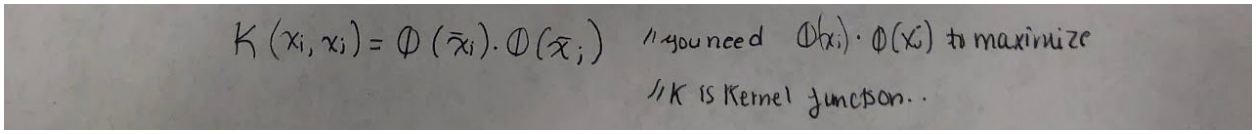


Maximizing width between support vector lines *//alpha is our hyperparameter*





Linear Kernel Function we use for our project…



In comparing our models to an open source library, we used the Scikit-Learn (sklearn) library for Python. For the purpose of testing the validity of our SVM, we compared our SVM models to sklearn's SVM models. Like our model, we used linear support vectors to provide a good basis of comparison. After doing so, we used polynomial support vectors to the second degree. Although a polynomial support vector may allow for better accuracy, we utilized a linear support vector because it provides for a better basis of comparison.

SKlearn

The SKLearn implementation that we used allows for kernel specification (we used linear and polynomial). Below are the functions and their descriptions:

`Fit (X, y, sample_weight)`

Paramteres:

X: the training vector; the number of samples; the number of feature

Y: class labels

Sample_weights: weights per sample; These are rescaled per sample; If the weight is high that point is considered more important.

This fuction returns itself as an object;;;

`Predict (x)`

This function performs the classification on the samples in X. It works on a one-class model (sample is either of the class or not) and returns +1, or -1 (return class labels for samples in X).

Results

Although we were unable to attain a high level of accuracy in classifying the data, we were able to attain good results using Sklearn. Our results were unexpected and did not follow typical trends in SVM classification. Below are our accuracy results:

SVM Results	*Linear*	*Quadratic*
Our models	52.88%	24.57%
Sklearn models	75.13%	TBD

First and foremost, our model had a very low level of accuracy in classifying data. It is also important to note that on UCI's data repository, many classification methods are discussed and analyzed. Of these methods, SVM was not one of them, and one of the reasons for this may be the fact that it did not perform well in classifying the data.

Typically, quadratic SVC is more effective in classification accuracy than the linear SVC because it is better fitted to the training data. In our models, the accuracy of the linear SVC was superior to the accuracy of the quadratic SVC. This may be due to overfitting, but there may have been some error with our model that adversely affected the results. Since the probability of randomly selecting the correct outcome is 50%, we would assume that our model should attain a higher outcome.

Sklearn's linear SVC attained a significantly higher accuracy than our linear SVC. We would assume that their quadratic SVC would attain an even higher accuracy. Unfortunately, since we are already late in submitting the project, we cannot wait the necessary time for the model to finish running. The quadratic kernel implementation has a much higher time

complexity than the linear kernel. That being said, the quadratic kernel is typically selected because it delivers more accurate and better fitted results.

In working with high dimensional data such as the Adult Data Set, we speculate that quadratic and even much higher degree polynomial kernel would perform better in optimizing predictions.

Working on this project emphasized the importance of cleaning data and formatting it into a manner in which it can be best worked with. Aside from constructing and implementing machine learning algorithms, other challenges in the project included feature selection, binning, and accuracy testing.

In constructing machine learning models themselves, we found unsatisfactory results, and by comparing our results with Sklearn, we gain insights into the magnitude and precision of the aspects involved in properly constructing optimal machine learning models. As opposed to viewing this as a setback, we see this as motivation - there is a lot to learn, and we are just scratching the surface, let alone venturing into uncharted territories - so with a scarcity of time, there is a great need for efficiency, precision, and optimization in our lives.

Works Cited

OpenCourseWare, MIT. “16. Learning: Support Vector Machines.” *YouTube*, YouTube, 10 Jan. 2014, www.youtube.com/watch?v=_PwhiWxHK8o.

“Figure 2f from: Irimia R, Gottschling M (2016) Taxonomic Revision of Rochefortia Sw. (Ehretiaceae, Boraginales). Biodiversity Data Journal 4: e7720. Https://Doi.org/10.3897/BDJ.4.e7720.” doi:10.3897/bdj.4.e7720.figure2f.

Yurtoğlu, Nadir.

“Http://Www.historystudies.net/Dergi//Birinci-Dunya-Savasinda-Bir-Asayis-Sorunu-Sebinkarahisar-Ermeni-isyani20181092a4a8f.Pdf.” *History Studies International Journal of History*, vol. 10, no. 7, 2018, pp. 241–264., doi:10.9737/hist.2018.658.