1 Ethics in Machine Learning

Lab Objective: *Machine learning algorithms can be extremely useful and convenient, but they can also have negative consequences. In this lab, we'll explore the impact people have on machine learning algorithms and some of the unintended effects of machine learning that can result.*

Introduction

Machine learning can be an extremely powerful tool in helping us interpret large datasets and then making decisions based on our data. A well-designed algorithm can use a dataset to train a model and identify patterns as well as make decisions with minimal human contact. As machine learning continues to advance and gain power and validity in the world, more and more datasets are being compiled to be trained on and then implemented into some kind of model. This implementation can be anything from predicting the next word when you are texting on your phone to typing at your computer to extremely powerful and accurate facial recognition software. Even though machine learning can be and is exceptionally useful, it has already demonstrated some drawbacks. These drawbacks have negative impacts that often outweigh the power and helpfulness of a well-designed machine learning model. In this lab we will be looking at a few of these drawbacks, as well as some ethical repercussions to help us understand how to be aware of and avoid them in our future endeavors.

Understanding Bias

To begin, we need to make a quick distinction about the term bias. Bias has many different meanings depending on the circumstance and field of study. From a mathematical perspective, *bias* is defined simply as a measure of the average error of an estimator, a statistical estimate of some quantity. In mathematical terms,

$$bias(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \hat{\theta}$$

where $\mathbb{E}$ is the expected value, θ is the term we are estimating, and $\hat{\theta}$ is the estimator. In machine learning, one of the goals is often to minimize this type of bias, which we will refer to as *statistical bias*.

This idea of minimizing bias also applies to other types of biases, including cognitive and data bias. These kinds of biases often result in predictions from our machine learning model being partial to a certain subset of the data. In this lab, we will investigate problems that involve several types of biases.

Achtung!

Not every type of bias has a clear or standard definition. Some types of biases even have different names. For instance, Wikipedia defines statistical bias more broadly,as “a systematic tendency in the process of data collection, which results in lopsided, misleading results". In this lab we have used common terms and definitions specific in the machine learning field but note that they are not universal. With sensitive topics like bias, it is important to be clear about the definition and meaning so that misunderstanding do not occur.

See `https://en.wikipedia.org/wiki/Bias#Statistical_biases` for Wikipedia’s definitions of different biases.

Statistical Bias and Variance

In machine learning there is a constant battle between statistical bias and variance. Often, the goal of a specific machine learning algorithm is to minimize error. Since the error of an algorithm can be described as the sum of irreducible error and reducible error (the sum of statistical bias squared and variance), this equates to minimizing statistical bias and variance.

$$Error = Bias^2 + Var + IrreducibleError$$

where if $g(x)$ is the function the model predicted and $f(x)$ is the actual target function,

$$Var(x) = \mathbb{E}[(g(x) - \mathbb{E}[g(x)])^2]$$

$$Bias(x) = \mathbb{E}[g(x)] - f(x).$$

By definition, a model with high variance means that small changes in the training set will result in large changes of the target function, so the model is overfitted. Low variance is just the opposite; changes in the training set will hardly affect the target function. Statistical bias, on the other hand, deals more specifically with the general form of the target function. High statistical bias implies that we are making large assumptions about the form of the target function, and small bias implies that we are making small assumptions about the form of the target function. Models with high bias are often classified as being underfitted, meaning it will not generalize well to any other data. Simply put, high bias assumes a model and tries to fit the data to that model, and low bias tries to fit a model to the data. Making the statistical bias smaller often makes the variance of the model go up and a smaller variance will result in a larger bias. The relationship between statistical bias and variance, or overfitting and underfitting, is inescapable in machine learning. In the end, the best algorithm is achieved by finding the middle ground.

Algorithm	Bias	Variance
Linear Regression	High	Low
Logistic Regression	High	Low
Decision Trees	Low	High
k-Nearest Neighbors	Low	High
Support Vector Machine	Low	High

Table 1.1: List of common machine learning algorithms whether they have low or high Bias and Variance.

A common example for evaluating this relationship is that of fitting a dataset to a polynomial. Based on our definitions of statistical bias and variance, we can conclude that as the degree of polynomial gets larger, the statistical bias decreases because the end result begins to depend more and more on the specific data points given to the training set.

We will be using the mean square error to calculate the error. The mean square error is defined as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{1.1}$$

Here Y_i is the given value and $\hat{Y}_i$ is the predicted value.

Consider the following example using `numpy` in fitting a third-degree polynomial to $cos(x)$.

```
>>> import numpy as np
>>> from sklearn.metrics import mean_squared_error as mse

>>> x = np.linspace(-1,1,25)
>>> degree = 3 # polynomial degree

# generate data to fit on
>>> y = np.cos(x)

# fit the polynomial
>>>poly_fit = np.polyfit(x, y, degree)

# evaluate the polynomial
>>> y_predicted = np.polyval(poly_fit, x)

# calculate the mean square error of the fit
>>> mse(y_predicted, y)
1.2075486783458322e-05

# plot the original and fitted data
>>> plt.plot(x,y,'r')
>>> plt.plot(x,y_predicted,'b')
```

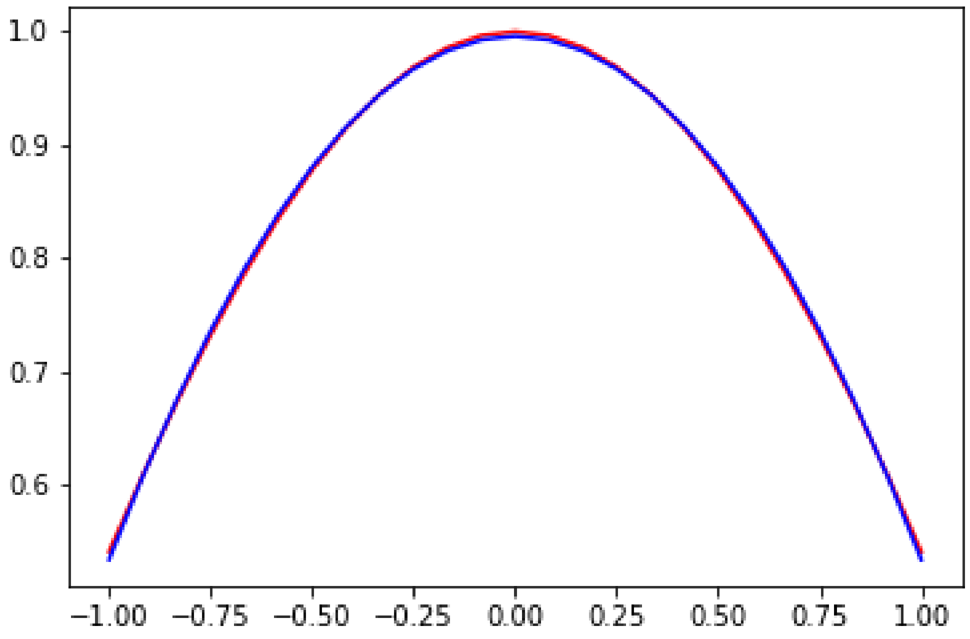

Figure 1.1: Plot of $cos(x)$ (red) and a third-degree polynomial approximation (blue).

To simulate the idea of generalization, we will generate 100 datasets. The better the algorithm performs across all the sets, the more we can assume that it will fit well on different datasets that were not in the training data.

Problem 1. Approximate $sin(\pi x)$ with polynomials. To do this, write a function that accepts as parameters `min_degree` and `max_degree`. Inside it, generate data using the provided `generate_sets()` function. `x_test` and `x_train` are 1-d arrays with the x-values needed to test and train. `y_test` and `y_train` are 100-d arrays, each subarray contains the y-values for one dataset.

For each dataset in `y_train`,

1. Fit a polynomial of each degree $d \in \{\text{min\_degree}, \ldots, \text{max\_degree}\}$, inclusive.
2. Use `np.polyeval()` to predict the values of `x_train` and `x_test` .
3. Calculate the `mean squared error` of the training and testing data.

Return the `mean squared error` of both the training and testing data for each dataset, as well as an array containing the predicted values for `x_test` for each degree and dataset, and the generated `x_test` values (needed to calculate the bias).

Plot the mean `test error` and the mean `train error` for each degree with the corresponding polynomial degree as the horizontal axis and compare with Figure 1.2.

Hint: The ndarrays outputted by your function should have the following shapes:

$$(100,9)(100,9)(100,9,7)(7,)$$

Now that we have the predictions and the error values, we can also evaluate the variance and bias. Recall that variance is calculated solely using the prediction values. In this case, the estimator g is the function generated when we used `numpy.polyfit()`.

Bias, on the other hand, is calculated using the estimator and the desired target function, which in this case is $\sin(\pi x)$. Using the equation for bias given above

$$Bias = (\texttt{mean}(y\_pred) - \sin(\pi x))^2$$

. For the returned `x_test` and `results_list` from `test_polyfits()`, we can calculate the mean variance across all the datasets on degree i using the method given below.

```
# Calculate the mean variance on all datasets of degree i
>>> mean_var = np.mean(np.var(results_list[:,i,:], axis=0))

# Calculate the mean bias on all datasets of degree i
>>> mean_bias = np.mean(np.square(np.mean(results_list[:,i,:], axis=0) - ↩
    target_function(x_test)))
```

Problem 2. Use the results of **??** to calculate the mean of the bias, variance, test error, and train error of each polynomial degree in range $\{0, \ldots, 8\}$. Plot the mean test error, variance, and bias with the corresponding polynomial degree as the horizontal axis. Make sure to include axis labels, a title, and plot legends for readability. Your plot should look like the right plot in Figure 1.2.

Create a single sample dataset with 500 samples using `generate_sets` with `num_datasets=1` and `num_samples=500`. Choose the polynomial of degree n that showed the smallest mean test error in the previous problems and fit your training set to that model.

Evaluate your results by plotting $sin(\pi x)$, the sample points, and the best fit polynomial. Include plot legends, display the `mean squared error` of the test set in the title of the plot, and display the degree of the polynomial as the label on the horizontal axis.

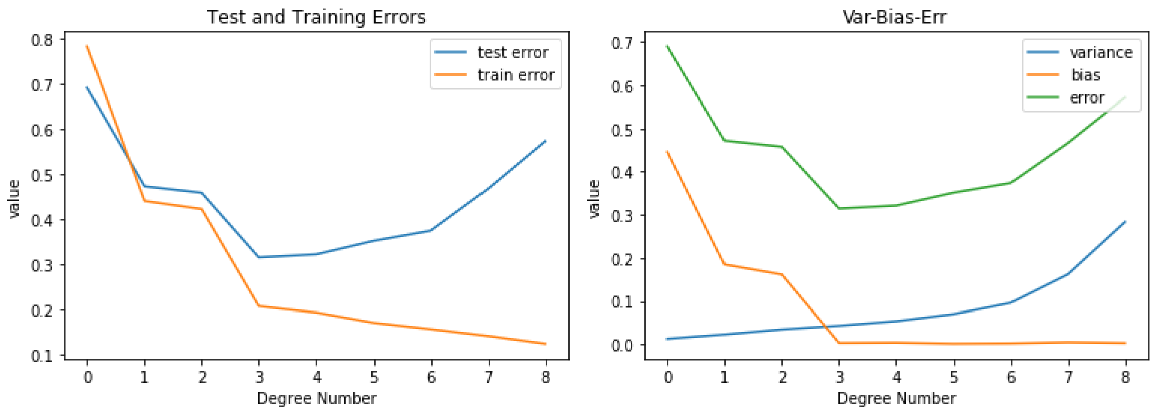

Figure 1.2

The way that you evaluated and chose the best algorithm for the given datasets is common in evaluating and choosing models to use in machine learning. If you continue to decrease the bias the training error will often continue to decrease, but there will be a point where the testing error starts to increase.

In Problem **??** we generated our own dataset, but if we are given a dataset from which we wanted to create a model, we can use *cross validation*, which is similar to what we did in the previous problem, to evaluate and choose the best model. *Cross validation* takes the given data and splits it into multiple training and testing sets. Using these new splits, we can then train and test on every one of these train-test-splits and evaluate the results. The model that showed the best performance across all of these splits is the one you will probably want to use, because it will perform better on datasets that were not in the training test. Computational complexity and size are other factors that could also influence your choice.

Though this kind of model evaluation can be both tedious and time consuming, but it is often the key to choosing the best model for a specific machine learning task as well as future model development. Even the smallest change in your model can result in a significant change in accuracy. Faulty data might look good initially but will not transfer over to different datasets very well. Evaluating the statistical bias and variance in conjuction with the error of your model will help you avoid these issues.

Measurement Bias

Measurement bias occurs when some of the data used as a variable is inaccurate. This could be because the equipment measuring the data can't detect small changes or the device measuring changes the data. One example is taking pictures on a camera that increases the brightness of the photo or has a spot on the lens.

Cancer detection is well-known machine learning problem. A model will train on images of cells, some of which are cancerous, and then predict whether new samples are cancerous. Some melanoma classification algorithms have been shown to predict melanoma better than trained physicians. This success has led to developers releasing software that allows a user to take picture and then the software will use a previously trained algorithm to predict the presence of melanoma. While this can be extremely useful and helpful, if there is a measurement error, such as a damaged camera lens, there can be issues in correct classification.

We will examine a model created to predict whether a specific skin lesion is melanomic. The files `melanoma_test.pkl` and `melanoma_modified.pkl` contain pandas *DataFrames* that will be used for testing the trained model which will be trained on `melanoma_train.pkl`. The file `melan oma.pkl` is a flattened array of a black and white image of a skin lesion. The other testing file `melanoma_modified.pkl` contains the same data, with an added black square to the image to simulate a faulty camera or damaged machine.

Problem 3. In this problem we will compare the differences between good data and faulty data in predicting if a person has melanoma. First, use `get_melanoma_data()` to get the relevant training and testing data. Notice that there are two different testing datasets, the *test* and *modified* datasets as explained above. Next, train a Random Forest model using SkLearn's RandomForestClassifier with `random_state=15`. After the model is trained, use the model to predict the `test` and `modified` datasets.

Compute the accuracy and the percentage of false positive results for both datasets. Display all four of these numbers in a `plt.bar` graph so that the reader can understand which ones are from the modified test data and which one is from the original image.

Write a few sentences explaining the results of the graph and how this can affect the usefulness of the model.

Sampling Bias

Sampling bias is a type of bias that results from the way the data, or a sample, is collected. More specifically, sampling bias is when the sample is collected in a way that results in some members of the intended population having a smaller sampling probability than others. This can happen when a sample is taken in a specific area that is not representative of the entire population. Sampling bias will result in many issues for prediction algorithms. In these next couple of sections and problems we will be focusing on only a few of them.

The Judicial System

Minority groups have a history of facing discrimination in many forms, including legal discrimination. Because of the power and usefulness of machine learning, predictive algorithms have already made their way into the judicial system as recidivism tools. One example is a machine learning algorithm that uses facial recognition with other factors, such as whether a defendant has a job and their education level, to produce a score called a risk assessment. The risk assessment score is then used to help determine things like sentence length, bond amount, and parole.

An analysis[1] showed that the algorithm had a 20% false positive rate for violent crime and of those predicted to commit another crime, about 61% did. It also showed that black defendants were 77% more likely to receive a higher risk for future violent crime and 45% more likely to receive a higher risk score for future crime of any kind. Many of these algorithms have not been independently evaluated for accuracy and racial bias, and the defendants are not allowed to see how their scores are calculated.

For Problems 4 and 5, we will be using `sentence_labeled.csv` and `sentence_unlabeled.csv`. The first will be used for training and testing the algorithm and the second will be used for prediction and analysis of the results. Each of these files contain a large sample of an even larger dataset

[1] `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`

containing information of incarcerated persons. The files have columns titled `AGE`, `RACE`, `OFFENSE`, and `FACILITY`. In addition, `sentence_labeled.csv` contains a column titled `SENTENCE YEARS`. This is the label that we will be training our model to predict, and it represents the sentence length in years. For simplicity, all sentences that were above 50 years were removed from the data.

We will be using sklearn's `RandomForestRegressor`, which is the continuous variable version of the `RandomForestClassifier`. To analyze the regressor we will be using the `R-squared` score. This scoring metric is a little more complicated, but it is used to help us determine the effectiveness of a specific regressor. If you wish to know more about it look at, `https://en.wikipedia.org/wiki/Coefficient_of_determination`. R-squared score is part of the sklearn metrics and takes two parameters, the actual labels and the predicted labels.

```
>>> from sklearn.metrics import r2_score

# compute R-squared score between actual (y_test) and predicted (y_hat)
>>> accuracy = r2_score(y_test,y_hat)
```

Problem 4. Identify the importance of the `RACE`, `FACILITY`, `AGE`, and `OFFENSE` features in `sentence_labeled.csv` in predicting `SENTENCE YEARS`. To do this, implement the following steps:

- Load the `sentence_labeled.csv`. Create the labels and split into test and training data, (use SkLearn's `test_train_split()`) with a 70/30 train/test split and `random_state=21`.
- Train the model using the `RandomForestRegressor` with `random_state=13`.
- Using the model, predict the labels for the test data. Calculate and print the R-squared score of the predicted test labels. It should be around $0.8 - 1.0$.
- Compute the feature importance (`model.feature_importances_` and create a descriptive bar plot with a column for each feature. Your results should look similar to Figure **??**.
- Remove `RACE` from the features. Retrain the model, predict the labels, compute the R-squared score and feature importance, and create a new bar plot.

  Write a few sentences about your results.

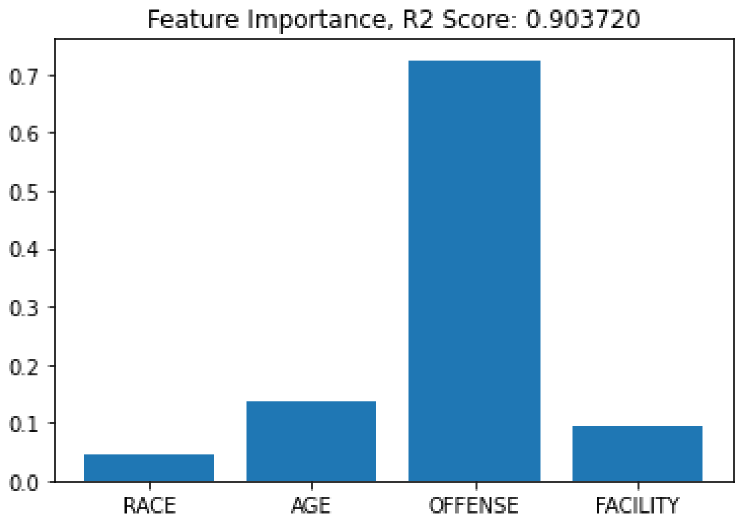

Figure 1.3

In this case, removing the $RACE$ feature will not eliminate prejudice because of the $FACILITY$ feature. The facilities that generally have longer sentences are those that are used to incarcerate the individuals of racial minority communities in the United States. More information on the facilities can be found in the `facility_map.p` file. It is common, that other less obvious features can also include sample bias in them. In other words, a column such as $FACILITY$, which does not necessarily indicate race, will also help to propagate negative prejudice into our machine learning models because it may disguise underlying characteristics. For example, since neighborhoods can be segregated, using zip codes in these areas can support a racial bias. It is extremely important to look at feature importance and compare results across demographics to prevent prejudice and discrimination.

Problem 5. Using the model with all four features, make predictions on the unlabeled data, `sentence_unlabeled.p`. Create histograms for the `SENTENCE YEARS` of black convicts for the following `OFFENSE` numbers: $27, 42, 95, 64$. Overlay the `SENTENCE YEARS` of the same `OFFENSE` for white convicts. Include plot legends and use 51 bins, each representing a single year from $0 - 50$. Use the `OFFENSE` number as a key to get its description using `offense_map.p` and use it as the title of the plot. See Figure 1.4 for an example.

Hint: Set the `hist` parameter `alpha` to a value which allows for you to see both plots, even if they overlap.

Note: These offenses have been chosen because the sample size was significant and almost equivalent for both black and white convicts.

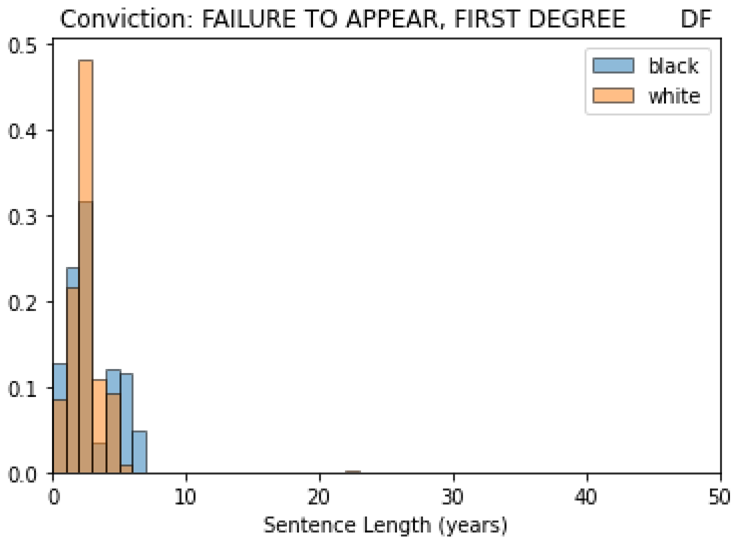

Figure 1.4: The percent of whites and blacks sentenced to lengths of time (in year).

Recognizing and Reducing Sampling Bias

Another example of sampling bias is found in resume learning. In 2014, Amazon's machine learning specialists started working on an algorithm that sorted through resumes and helped choose the applicants who would qualify for an interview. Given the large number of application Amazon receives and the time needed to sort through resumes, this seemed like a great idea. However, the next year, the company discovered that the algorithm was prejudiced against women[2]. So a woman was significantly less likely to be given an interview than a man, even though both applicants had more or less the same qualifications.

This is an additional example of sampling bias, because the data Amazon used to train their model on was data from previously hired employees over the previous ten years. Because the tech industry is a male dominated industry, the alogrithm favored patterns found on male resumes. This sample, based on imbalanced data, resulted in a machine learning algorithm that often chose men over women with the same relevant qualifications.

The typical machine learning algorithm for resume sorting goes as follows: The first part uses natural language processing techniques to pick keywords off each resume to make sorting easier. A model is trained on the keywords to determine which keywords are most important in determining whether a candidate successfully gets hired. Going back to the Amazon example, their model learned keywords and phrases that are were common on male applicants' resumes, such as `executed` and `captured`, and counted them as important, even though the words and phrases were irrelevant to the actual job.

For the next problem we will be considering a similar scenario. A group of veterans with special programming and development skills came together and formed a software engineering company. They decided that their platform would be structured around fairness for all. Because of their specific advertising and present demographic, they received almost exclusively veteran applicants. Accommodations for disabled veterans were in place and posed no problems for those wanting to apply.

[2]https://reuters.com/article/us-amazon-com-jobs-automation-insight

The company grew and began receiving more skilled applicants outside of their specific veteran demographic. To reduce the time it took to sort through the growing stack of resumes, they decided to create a model that could do the preliminary sorting for them. Using previous hiring data, they created an algorithm that selected keywords that were common on applicants and simplified all the previous and current applicants based on those key words. After this, they created the model to make the prediction that they wanted.

The data is located in three files; `skills.txt`, `resumes_train.csv`, and `resumes_test.csv`. The first file, `skills.txt` contains the key words the algorithm identified as important on the resumes. The second file, `resumes_train.csv`, contains the training data. It is a pandas *DataFrame* and contains a list of resumes in csv form, where each column is labeled with a keyword from the resume with corresponding value 0 (keyword was not on resume) or 1 (keyword was on resume). The first column is `Interview`, which indicates if the resume owner received an interview. `resumes_test.csv` is the new batch of resumes, so it is similar to `resumes_train.csv` but without a column to indicate whether a person received an interview or not.

Problem 6. Analyze the algorithm that the company created. The first steps of identifying key words, simplifying the resumes and model creation have all been done for you. To get the accuracy and results of the model call,

```
# get the accuracy and the results of the algorithm
>>> accuracy, results = resume_model("skills.txt")
```

This function will return a cross validation mean accuracy score called `accuracy`, and a pandas `DataFrame`, called `results`. `results` contains the training data. Each row represents an applicant and the columns represent a keyword that could appear on the applicant's resume. The last column, `'Interview'`, is the model's prediction of whether that applicant received an interview.

Create a function called `get_percentages()` that accepts the `results` and calculates and returns the following.

- The percentage of applicants the model predicted to receive an interview.
- The percentage of women applicants predicted to receive an interview.
- The percentage of veteran, women applicants predicted to receive an interview.
- The percentage of non-veteran, women applicants predicted to receive an interview.

Finally, use a `plt.bar` graph to display the cross validation accuracy score, the percentage of people who received an interview, the percentage of women that received an interview, the percentage of women who were veterans that received an interview, and the percentage of women who are not veterans that received an interview. Compare your graph with Figure 1.5.

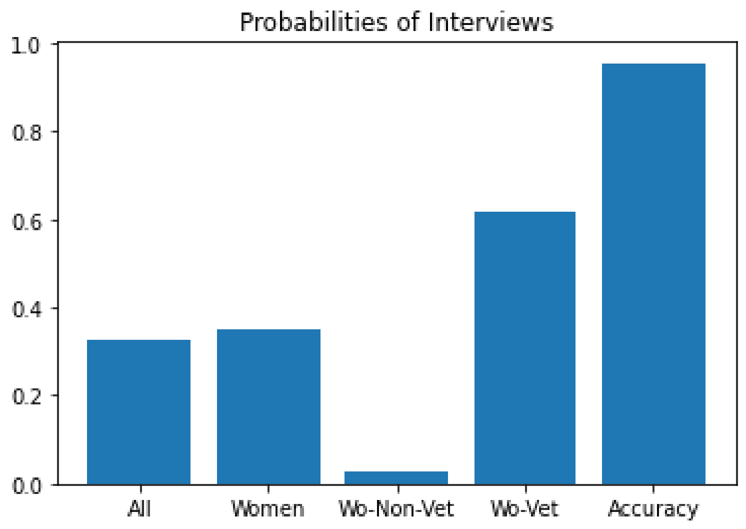

Figure 1.5: Probability different demographics would receive an interview.

If we only looked at three of the results: the percentage of applicants that recieved an interview, the percentage of women that received an interview, and the accuracy, we would think that the model performed well. The percentage of women who received an interview was similar to the percentage of total people who got an interview. However, if you look at more specific results, like the percentage of women veterans that got the interview versus the percentage of non-veteran women that received an interview, the algorithm shows some intense favoritism to veterans.

As discussed previously, the model to select keywords was trained on the company's previous data. This data was gathered from a time when they previously received applications from and hired exclusively veterans. Some of these key words in *skills.txt* are not relevant for a software engineer, but the algorithm deemed them important because of the frequency of their appearance in the training data. The training data did not always represent what the employers were actually looking for.

Discussion on Ethics

In this lab, we discussed many important techniques for examining and determining multiple types of bias in machine learning algorithms. *Statistical bias* and *variance* can be optimized and *measurement error* can be avoided, but other kinds of cognitive and social bias are more nuanced. Being aware of these issues is important as machine learning becomes ingrained in how the world and its machines function. We will mention three ethical concerns regarding machine learning that were more or less a part of this lab. Be aware that this list of three does not contain all possible ethical questions when it comes to machine learning.

The first concern is of the fairness of the model. We examined this idea in both the resume and the incarceration problems. Data based on previous prejudice or sampling bias may be replicated in your algorithm, propagating error and resulting in unintentional discrimination. Avoiding this problem requires conscious effort and a proper understanding of the sources of your data and the impact of the model.

The second concern is dehumanization. Machine learning can take humans out of many important decision-making processes, which can be a good thing if the algorithms are extremely well designed, but it also poses a significant risk if the model contains any amount of accidental bias or discrimination, as in the incarceration problem.

The final idea is consent. It is important to consider what data the model uses, whose data it is, and if it was gathered appropriately. Privacy and permission must be respected. This is an important concern to consider before beginning a project and while considering potential effects of the analysis.

There is not a single answer to any of these questions that will please everyone, such is the nature of ethics, but before implementing a machine learning algorithm, consider what you are doing and how it might affect those around you. As this issue gains awareness, people are starting to create tools to identify and mitigate bias and discrimination.

- AI Fairness 360[3]: Created by IBM, it is a well-known open-source package that not only identifies and mitigates discrimination and bias.
- Skater[4]: Developed by Oracle, Skater can be used on complex algorithms and black-box models to detect bias.
- Audit-AI[5]: A python library that can integrate with Pandas an SkLearn to measure and mitigate discriminatory patterns.

Other python packages for identifying bias include `FairML`, `aequitas`, `fairNN`, and `parity-fairness`.

For more information on fairness, accountability, transparency, and ethics in machine learning, visit `https://www.microsoft.com/en-us/research/theme/fate/`.

[3]`https://aif360.mybluemix.net/`
[4]`https://pypi.org/project/skater/`
[5]`https://github.com/pymetrics/audit-ai`