Statistical Modelling: Facial Recognition

How effective are statistical models alongside dimension reduction at identifying images of dogs?

SUBJECT: MATHEMATICS

WORD COUNT: 3757

TABLE OF CONTENTS

Chapter 1: Introduction

Logistic Regression, k-Nearest Neighbors & Linear Perception are all models that can be used to classify entities. These models use high levels of mathematics to determine what classification is assigned to an entity given its features. This paper seeks to explore these different statistical models and observe how they can be used in conjunction with dimension reduction techniques to identify images, more specifically ones of dogs. Due to this, this paper's research question is:

HOW EFFECTIVE ARE STATISTICAL MODELS ALONGSIDE DIMENSION REDUCTION AT IDENTIFYING IMAGES OF DOGS?

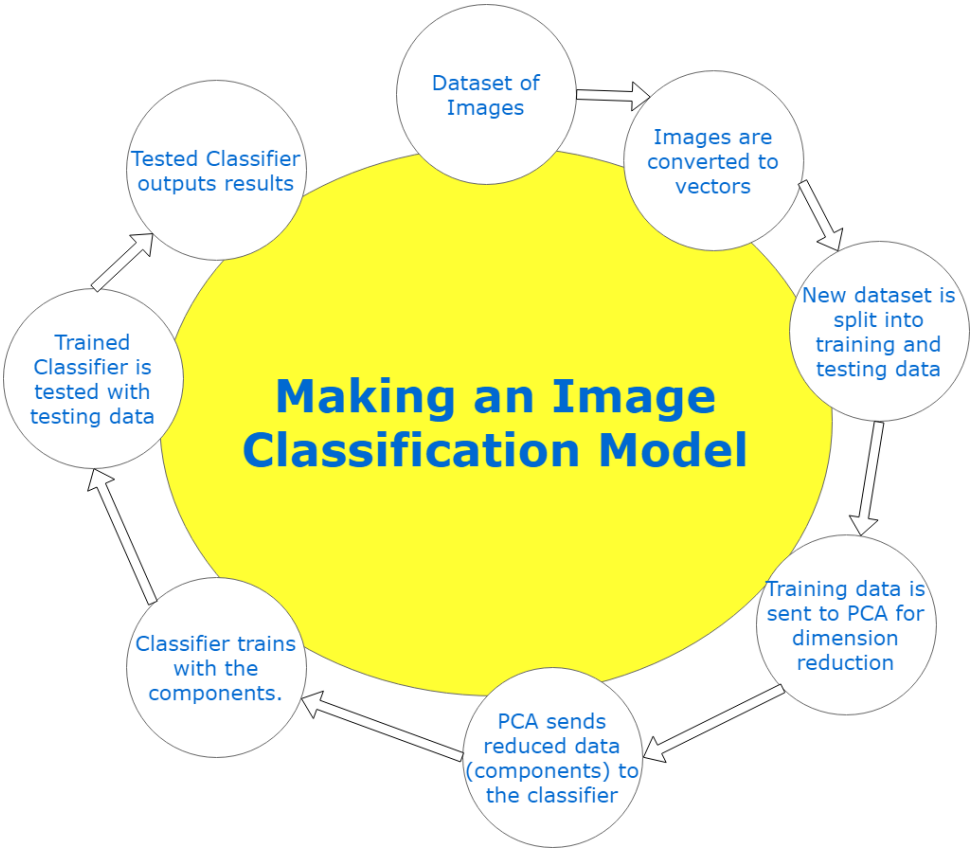
This paper seeks to classify images of dogs due to the pre-existing threat of rabies; even with yearly vaccination from 1977 to 1981, 102 rabies cases were reported (Alonge and Abu). In more recent years (2013-2015), the reported rabies to dog bite ratio in Ghana was 3:1000, however in a country where there are 172 dog bites per 100,000 people, dogs are potentially problems for one's wellbeing.

Even though there are many statistical models, the models that will be investigated are **k-Nearest Neighbors (KNN)**, **Logistic Regression (LR)** and **Linear Perception (LP)**. This is because each one represents a different model type; **KNN** is a non-parametric model, **LR** is a parametric model and **LP** is an artificial neural network. These variations allow for an effective evaluation of the different models.

Principal Component Analysis (PCA), will be used as the dimension reduction technique due to its sophisticated method of generating features from images containing large numbers of parameters.

In this paper, Python 3 will be used to create and test the model. Python 3 was picked over the other programming languages as a result of its elegant and simple syntax which can easily run complex machine learning programs.

This research is worthy of investigation because its results can be applied in many other disciplines with different subjects of investigation (eg. Identifying images of cancerous lungs to improve medical diagnostics). It also assesses the strength of these models in performing complex tasks and gives insight into the reliability of these models.



Source: Author 16/10/2018

*The dataset will be split into the training data and testing data with 75% training and 25% testing.

Chapter 2: Image Classifiers

L. Sirovich and M. Kirby, first came up with the idea that pictures of faces can be represented as eigenfaces which can be used to classify faces. Eigenfaces are sets of eigenvectors which are any vectors that satisfy the equation

$$Ax = \lambda x$$

Where in this case, A is an image represented as a square matrix, x is a column vector containing eigenvectors and λ is a scalar value.

If like terms are grouped:

$$Ax - \lambda x = 0$$

A unit matrix I is introduced so that λ can be subtracted from I. It is important to note that $x \neq 0$ because so that the solution will be non-trivial (Stroud and Dexter 509-10).

$$(A - \lambda I)x = 0$$

Since $x \neq 0$, the only solution to the equation is

$$|A - \lambda I| = 0$$

$$|A - \lambda I| = \begin{vmatrix} (a_{11} - \lambda) & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & (a_{nn} - \lambda) \end{vmatrix}$$

The determinant of the equation gives a polynomial of degree n whose solutions are the eigenvalues. And by substituting these eigenvalues, the different eigenvectors can be calculated.

For example, given an image $A = \begin{pmatrix} 4 & 1 \\ 3 & 2 \end{pmatrix}$

$$|A - \lambda I| = \begin{pmatrix} (4-\lambda) & 1 \\ 3 & (2-\lambda) \end{pmatrix}$$

The equating the determinant of $|A - \lambda I|$ to 0 gives

$$(4 - \lambda)(2 - \lambda) - 3 = 0$$

$$\lambda^2 - 6\lambda + 8 - 3 = 0$$

$$\lambda^2 - 6\lambda + 5 = 0$$

$$\lambda_1 = 1;\ \lambda_2 = 5$$

This means that the image ***A*** has eigenvalues of 5 and 1.

By substituting $\lambda = 1$,

$$Ax = \lambda x$$

$$\begin{pmatrix} 4 & 1 \\ 3 & 2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 1 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\begin{pmatrix} 4 & 1 \\ 3 & 2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Resulting in the same two equations,

$$4x_1 + 1x_2 = 1x_1$$

$$x_2 = -3x_1$$

$$3x_1 + 2x_2 = x_2$$

$$3x_1 = -x_2$$

$$-3x_1 = x_2$$

This means that $x_1 = k$ and $x_2 = -3k$ producing and eigenvector of $\begin{pmatrix} k \\ -3k \end{pmatrix}$ which in its simplest terms is, $\begin{pmatrix} 1 \\ -3 \end{pmatrix}$.

By substituting $\lambda = 5$,

$$Ax = \lambda x$$

$$\begin{pmatrix} 4 & 1 \\ 3 & 2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 5 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\begin{pmatrix} 4 & 1 \\ 3 & 2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 5x_1 \\ 5x_2 \end{pmatrix}$$

Resulting in the same two equations,

$$4x_1 + 1x_2 = 5x_1$$

$$x_2 = x_1$$

$$3x_1 + 2x_2 = 5x_2$$

$$3x_1 = 3x_2$$

$$x_1 = x_2$$

This means that $x_1 = k$ and $x_2 = k$ producing and eigenvector of $\begin{pmatrix} k \\ k \end{pmatrix}$ which in its simplest terms is, $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

From this the calculated eigenvalues with their corresponding eigenvectors are:

$$\lambda = 1\ , x = \begin{pmatrix} 1 \\ -3 \end{pmatrix}$$

$$\lambda = 5\ , x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Turk and Pentland moved further to say that these eigenvalues & eigenvectors can be used in image recognition, and proposed that they could use **PCA** on a training set of images to produce a smaller number of eigenvalues & eigenvectors, which would be used to classify other images in the testing data. (Turk and Pentland 588)

For example

Take a set of m images as the training data

{ I_1 , I_2 , I_3, I_4, I_5 ..... I_m}

Each image is made up of *n* pixels

$I_i = \{ x_1 , x_2 , x_3, x_4, x_5 \ldots\ldots\ldots x_n\}$

PCA would use these images to find **k** eigenvectors that best describe all images.

$\{y_1, y_2, y_3, y_4, y_5 \ldots\ldots\ldots\ldots y_k\}$ where $\mathbf{k < n}$

Then each image will be described with the eigenvectors and their respective eigenvalues.

$\mathbf{I}_1 = \{0.67\ y_1 , 0.28\ y_2 , 0.37\ y_3 , 0.13\ y_4, 0.87\ y_5 \ldots\ldots\ldots\ldots 0.43 y_k\}$

The coefficients of the eigenvectors are the calculated eigenvalues of the given image.

These eigenvalues would be given to the classification models. These classifiers will then use their various methods to classify the given images.

2.2 Data Collection

A dataset is necessary to be able to train and test the model. The dataset that was used consisted of 1000 images of dogs and cats (500 dogs and 500 cats), to ensure that the dataset was trustworthy it was taken from Kaggle, a well renowned data science website. Below are some sample images from the dataset.

SAMPLE DATASET IMAGES

Source: Kaggle 09/10/2018

With the 1000 images, the dataset was split into two: **training data** and **testing data** in the ratio 3:1 (750 training images and 250 testing images).

Chapter 3: Research and Analysis

3.1 Dimension Reduction: Principal Component Analysis

Images are combinations of pixels with each pixel having its own pixel intensity value. In this research the used images were changed to grayscale, so that each pixel had a value between 0 - 255 representing its pixel intensity.

Since images are combinations of pixels, grayscale images can be represented as either matrices or column vectors where each position is a pixel's intensity. This forms the coordinates which are given to PCA to find the principal components.

PCA takes the image dataset with each image containing the same number of pixels and returns a smaller or equal number of eigenvectors that best describes the variation in the dataset. (Abdi and Williams 437)

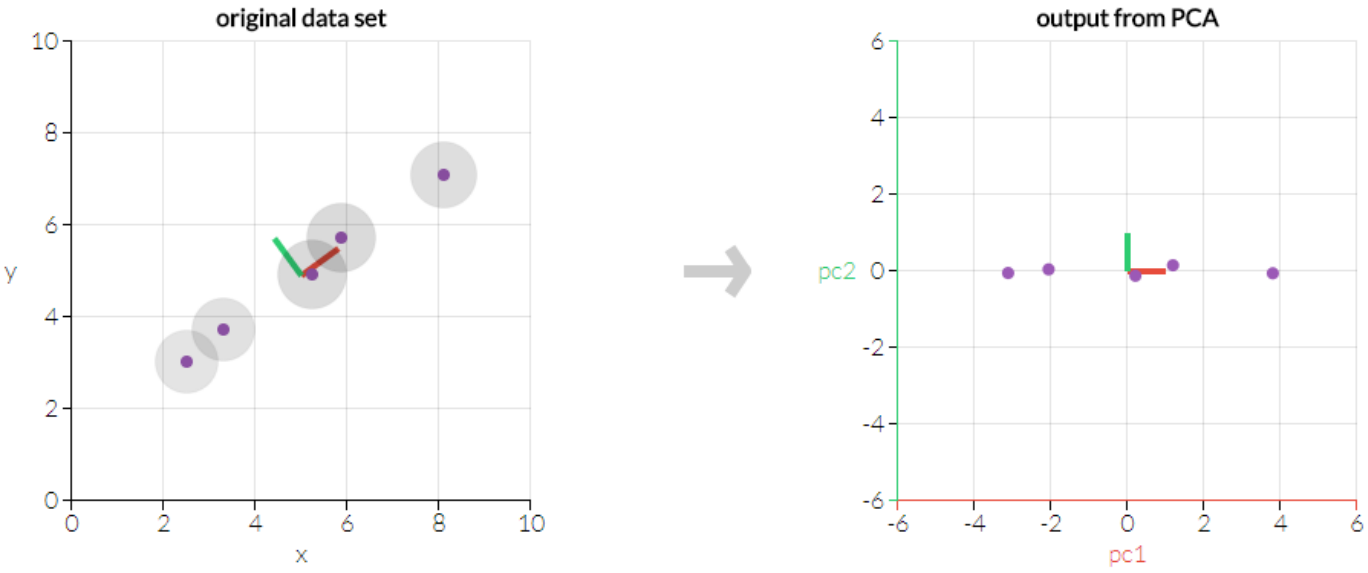
Each image is mapped onto the returned eigenvectors using the equation

$$Ax = \lambda x$$

Where A is the image represented as a square matrix, and λ is the mapped image as a column vector.

Each eigenvector is called a "Principal Component" and they are arranged in order of significance e.g. the first principal component describes greater variance than the second principal component and so on.

If the principle components each placed on an axis instead of the former parameters, then the relationship between the components will be more visible as shown below:



Source: Powell and Lehe 2015

Note: pc1 and pc2 mean first and second principal component respectively

How PCA determines the eigenvectors

PCA plots the training data and draws a straight line passing through the center point of all the training data (this would be the average point of all the data) and projects the training data onto the straight line to make it a point on the line. It then finds the sum of each projected point's squared distance from the average point (the distances are squared to get the magnitude of each point to best depict the variance) and stores it.

Then PCA alters the gradient of the line and maps the training data onto the new line and finds the sum of squared distances and stores it. PCA finds the best fit line by maximizing the sum of the squared distances from projected points to the origin. This is because the sum of squared distances represents the variance for the points and the greater the variance, the easier it is to differentiate between images.

After it finds the best fit line with the projected points, it makes that line its first component and returns the points in terms of that line as the first set of parameters. PCA will repeat this process finding the next best fit line and making that the second component and so on.

These lines are then transformed into eigenvectors which are outputted by PCA.

(StatQuest).

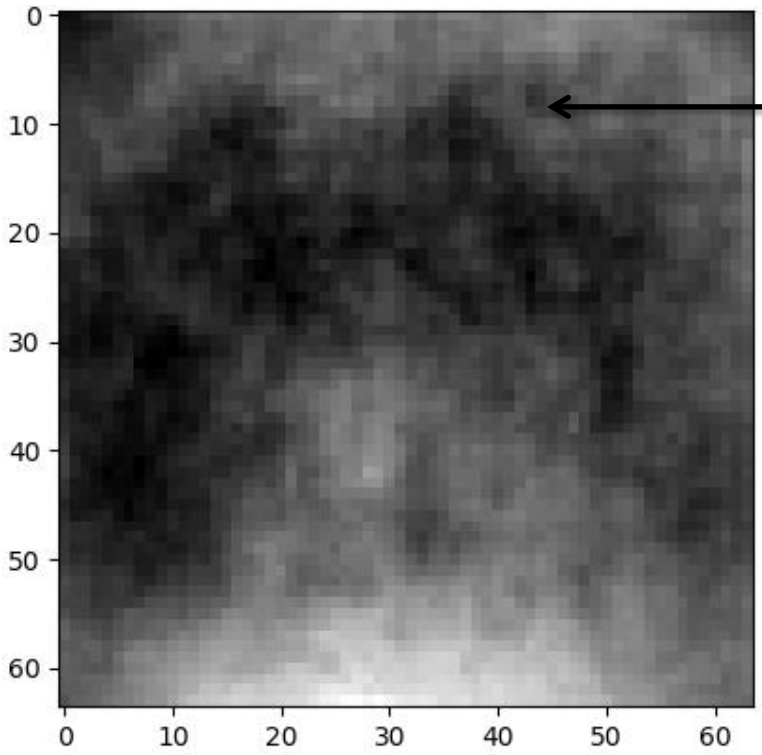
Mathematics for calculating eigenvectors and eigenvalues

Given m pixels from each of the n images the following steps find the principal components for PCA.

1) Find the "mean face", the average of all the training data images

Let M represent the mean face and X_i represent each image in the training data

$$\vec{M} = \frac{1}{n} \times \sum_{i=1}^{n} X_i$$



The mean face from the dataset represented as an image

Source: Author, 25/10/2018

2) Center the training data so that the mean becomes the origin

Let B be an m x n matrix

$$\mathrm{B} = [\vec{X}_1 - \vec{M} | \ldots\ldots\ldots | \vec{X}_\mathrm{n} - \vec{M}]$$

3) Create the covariance matrix (a matrix whose diagonal shows the correlation between the vectors in the matrix and outputs the variance of each variable)

Let $C_{x,y}$ be the m x m covariance matrix

$$\mathbf{C_{x,y}} = \frac{\mathbf{1}}{\boldsymbol{n-1}} \boldsymbol{BB^T}$$

Where x and y are parameters

Rules

1) If $x = y = k$ then the diagonal matrix outputs the variance of the *k* th axis.
2) If $x = k$, $y = j$ and $k \neq j$ then diagonal matrix outputs the covariance of the j th and *k* th axis

(Jauregui)

B^T represents the transposed matrix B, where the rows become the columns and the columns become the rows. For example $[0,1,2]^{\mathrm{T}} = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$

<u>Example</u>

Given 3 images X_1, X_2, X_3 with a mean $\vec{M}$

$$\vec{X}_1 = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{m-1} \\ a_m \end{bmatrix}, \vec{X}_2 = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{m-1} \\ b_m \end{bmatrix}, \vec{X}_3 = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{m-1} \\ c_m \end{bmatrix}, \vec{M} = \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{m-1} \\ M_m \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} a_1 - M_1 & b_1 - M_1 & c_1 - M_1 \\ a_2 - M_2 & b_2 - M_2 & c_2 - M_2 \\ \vdots & \vdots & \vdots \\ a_{m-1} - M_{m-1} & b_{m-1} - M_{m-1} & c_{m-1} - M_{m-1} \\ a_m - M_m & b_m - M_m & c_m - M_m \end{bmatrix}$$

These images would be plotted on an m dimesional graph

$$C_{1,1} = \frac{1}{3-1}((a_1 - M_1)^2 + (b_1 - M_1)^2 + (c_1 - M_1)^2)$$

This outputs the variance of the first axis (if in 2D it would be the variance of the x-axis)

This means that the total variance can be found by summing the variances of all the axes.

$$Total\ variance = \sum_{i=1}^{m} \boldsymbol{C_{i,i}}$$

4) Apply the Spectral Theorem

Transposing **B** gives the same result; **B** is **symmetric**

Spectral Theorem states that:

Given **B** an m x n matrix

If $\mathbf{B}^T = \mathbf{B}$, then there exists real numbers or eigenvalues $\lambda_1, \lambda_2, \ldots\ldots, \lambda_n$ and eigenvectors $v_1, v_2, \ldots\ldots, v_n$ that satisfies the following equation.

$$\mathbf{B}v = \lambda v$$

(Gockenbach)

This is the same equation as shown in Chapter 2 **Image Classifiers**, page 5. However here the input is the adjusted **B**

There is a diagonal matrix D which is the main diagonal of **B** and an orthogonal matrix U whose columns represent the eigenvectors. Both D and U can be multiplied to produce A:

$$\mathbf{B} = \mathbf{UDU^T}$$

This is referred to as the special decomposition of and by decomposing the eigenvalues (D) and eigenvectors (U) can be obtained to send to the classifiers.

Substituting this gives:

$$\mathbf{C_{x,y}} = \frac{\mathbf{1}}{\boldsymbol{n-1}}\boldsymbol{BB^T}$$

$$\mathbf{C_{x,y}} = \frac{\mathbf{1}}{\boldsymbol{n-1}}(\mathbf{UDU^T})(\mathbf{UDU^T})^{\boldsymbol{T}}$$

Since D is a diagonal matrix, transposing it gives the same result.

$$\mathbf{C_{x,y}} = \frac{\mathbf{1}}{\boldsymbol{n-1}}(\mathbf{UDU^T})(\mathbf{UDU^T})^{\boldsymbol{T}}$$

$$\mathbf{C_{x,y}} = \frac{\mathbf{1}}{\boldsymbol{n-1}}(\mathbf{UDU^T})(\mathbf{U^TDU})$$

$$\mathbf{C_{x,y}} = \mathbf{U}\frac{\boldsymbol{D^2}}{\boldsymbol{n-1}}\mathbf{U^T}$$

$$\boldsymbol{E} = \boldsymbol{argmax}(\mathbf{C_{x,y}})$$

The eigenvectors that produce E are used as the principal components.

Model Eigenfaces

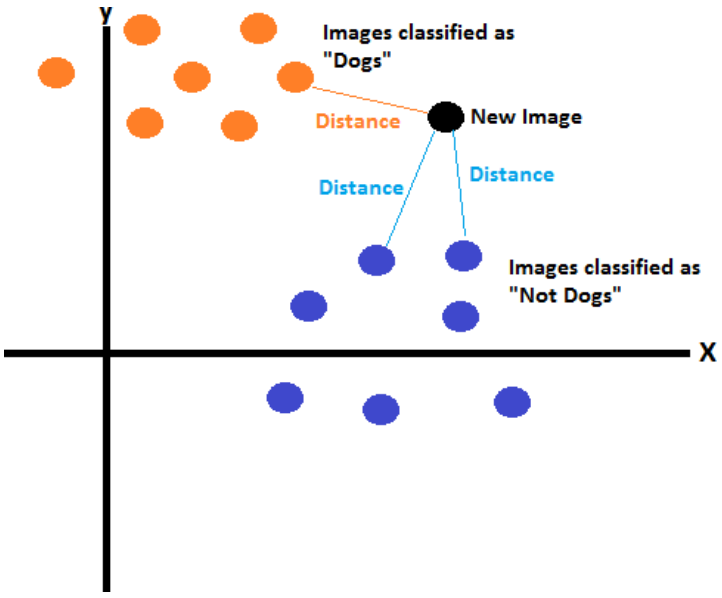
The dataset consists of 1000 64x64 images of dogs and cats (500 dogs and 500 cats). Since the images are 64x64 they each have 4096 pixels which can be used as the parameters that

describe each image. This means that all of the 1000 images can be represented as 4096x1 vectors. After passing these through PCA, the total number of eigenvectors that explain 95% of the variance in the training data was 238. This means that the dimensions have been reduced by 94% (from 4096 to 238).

3.2 Classification Model 1: K-Nearest Neighbors

K-Nearest Neighbors (k-NN) is a non-parametric method of classification (Zakka). After the images are mapped onto the principal components, their eigenvalues are sent to k-NN as data to be used for both training and testing. K-NN plots the training data on a graph, and when it wants to classify a new image it finds the k nearest neighbors and uses a "majority vote" rule to determine which classification to give the new image.

When a new image is inputted to be classified, it is mapped onto the principal component and its corresponding eigenvalues to are used to plot the image on the graph:



Source: Author, 09/11/2018

In order to find the distance between two points, k-NN has different methods such as Hamming Distance, Euclidean Distance and so on. However, this paper will only use the Euclidean distance as means of measuring the difference between two points due to limitations of the program.

Given a new image **I** with **m** eigenvalues = $\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{m-1} \\ a_m \end{bmatrix}$,

k-NN will calculate the Euclidean Distance from each point by using the equation:

D(**I**, **X**) = $\sqrt{(x_1 - a_1)^2 + (x_2 - a_2)^2 + \cdots + (x_m - a_m)^2}$

Where X is any image in the dataset with **m** eigenvalues = $\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{m-1} \\ x_m \end{bmatrix}$

k-NN will calculate the Euclidean distance of the new image to all of the training data images, then it will find the closest k training data images or "neighbors". Each of the **k** neighbors will vote for their own classification (eg. An image of a dog will vote that the new image is a dog) and the classification with the most votes will be given to the new image.

Since the value of **k** can vary, the optimum value for **k** can be found. Normally, **k** = 1 is the favoured value (Zakka). This is because when **k** = 1 it finds the closest training data image and classifies itself as the image's classification (dog or not a dog). The optimum number of neighbors will be derived finding a relationship between the number of neighbors and the accuracy of the model.

3.3 Classification Model 2: Logistic Regression

This is a model that uses a parametric equation to classify images.

Take this equation to be the equation for accurately identifying a dog:

$$Y = f(X)$$

Where,

f(x) = b+ $b_1X_1 + b_2X_2 + b_3X_3 + \ldots + b_mX_m$, X_i is a principal component/ eigenvalue and there are **m** given principal components in total

Multiple regression seeks to produce an estimate parametric equation that can predict the animal in an image with as little error as possible.

$$\hat{Y} = \hat{f}(X) + e$$

Where,

"e" represents the error between Y and $\hat{Y}$, this can be calculated by $\hat{Y}$ - Y.

$\hat{f}(X) = \hat{b} + \hat{b}_1X_1 + \hat{b}_2X_2 + \hat{b}_3X_3 + \ldots + \hat{b}_mX_m$

<u>Calculating Coefficients</u> (CodeEmporium):

1) Put the equation in vector form

With **n** given principal components and **k** images for training data.

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_{n-1} \\ Y_n \end{bmatrix}, X + e = \begin{bmatrix} 1 + & X_{1,1} + & \ldots + & X_{1,n} + & e_1 \\ 1 + & X_{2,1} + & \cdots + & X_{2,n} + & e_2 \\ \ldots + & \cdots + & \cdots + & \cdots + & \cdots \\ 1 + & X_{n-1,1} + & \cdots + & X_{3,n} + & e_3 \\ 1 + & X_{k,1} + & \cdots + & X_{k,n} + & e_k \end{bmatrix}, B = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{n-1} \\ e_n \end{bmatrix}$$

The equation: $Y = BX + e$ **or** $\hat{Y} = \hat{B}X$ can be formed

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_{n-1} \\ Y_n \end{bmatrix} = \begin{bmatrix} B_0 + & B_1X_{1,1} + & \dots + & B_nX_{1,n} + & e_1 \\ B_0 + & B_1X_{2,1} + & \cdots + & B_nX_{2,n} + & e_2 \\ \cdots + & \cdots + & \cdots + & \cdots + & \cdots \\ B_0 + & B_1X_{n-1,1} + & \cdots + & B_nX_{3,n} + & e_3 \\ B_0 + & B_1X_{k,1} + & \cdots + & B_nX_{k,n} + & e_k \end{bmatrix}$$

2) Minimize the sum of squared errors

As previously mentioned e is Y - $\hat{Y}$ and this represents the models error. The error term is squared to get the magnitude of the error.

$$\text{Sum of Squared Errors} = \sum_{i=1}^{n} (\text{e}_\text{i})^2$$

This is equal to $\boldsymbol{e^T}$**e** which can be minimized through the use of some matrix differentiation rules.

With **x** = k x 1 matrix and **A** = n x k matrix both matrices perpendicular to each other

Rule 1)

If Y = **A**

Then $\frac{\partial y}{\partial x}$ = **0**

Rule 2)

If Y = **Ax**

Then $\frac{\partial y}{\partial x}$ = **A**

Rule 3)

If Y = **xA**

Then $\frac{\partial y}{\partial x}$ = $\mathbf{A^T}$

Rule 4)

If Y = $\mathbf{x^TAx}$

Then $\frac{\partial y}{\partial x}$ = $\mathbf{2x^TA}$

$\boldsymbol{e^T e} = (Y - \hat{Y})^T(Y - \hat{Y})$

$= (Y^T - (X\hat{B})^T)(Y - X\hat{B})$

$= (Y^T - \hat{B}^T X^T)(Y - X\hat{B})$ This is because $(X\hat{B})^T = \hat{B}^T X^T$

$= Y^T Y - Y^T X\hat{B} - \hat{B}X^T Y + \hat{B}^T X^T X\hat{B}$

To minimize the sum of squared errors it is differentiated and equated the differential to 0 to find the value of $\hat{B}$ that minimizes the sum.

Differentiate with respect to $\hat{B}$

$$\frac{\partial(\boldsymbol{e^T e})}{\partial \hat{B}} = \frac{\partial(\boldsymbol{Y^T Y})}{\partial \hat{B}} - \frac{\partial(Y^T X\hat{B})}{\partial \hat{B}} - \frac{\partial(\hat{B}X^T Y)}{\partial \hat{B}} + \frac{\partial(\hat{B}^T X^T X\hat{B})}{\partial \hat{B}} = 0$$

$0 = 0 - Y^T X - Y^T X + 2\hat{B}^T X^T X$

This is because of the matrix differential rules. Rule 1 for the first differential, Rule 2 for the second, Rule 2 and 3 for the third and Rule 4 for the last one.

$$2Y^T X = 2\hat{B}^T X^T X$$

$$Y^T X = \hat{B}^T X^T X$$

$$\hat{B}^T X^T X = Y^T X$$

$$\hat{B}^T = Y^T X(X^T X)^{-1}$$

The coefficients for multiple regression is calculated from the formula above. This creates the model to identify whether an image is a dog or not and give a probability for each. The largest is chosen and that classification is given.

Logistic Regression: Calculating Probability

In order to calculate probability we need to make the model output values from 0 to 1. This is the problem with just using linear regression as the model; linear regression outputs values from -∞ to ∞. The linear line is scaled between 0 and 1 with the use of a sigmoid function. A sigmoid function is an "S" shaped curve that can scale the linear equation (CodeEmporium). It has the form:

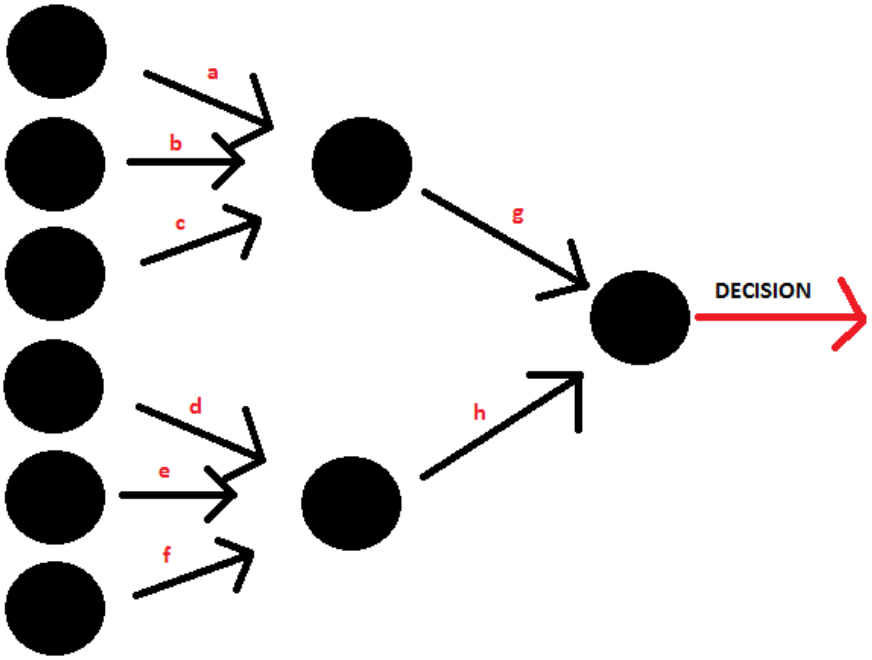
$$S(x) = \frac{e^x}{1 + e^x}$$

$S(\hat{Y})$ scales the line into a sigmoid function. And by inputting the eigenvalues gives the probability of the image being a dog. If it is greater than 0.5 then the image is classified as a dog, otherwise it is classified as not a dog.

3.4 Classification Model 3: Linear Perception Classifier (LP Classifier)

Linear Perception (LP) is an artificial neural network that makes its classification decisions based on the outputs of the neural network model. This method attempts to imitate the human brain by first processing inputs before making decisions.

An artificial neural network models the human brain, consisting of large numbers of neuron models that take in inputs from other neurons and output a single numerical value (represented by a,b,c,d,e,f,g,h below).



SOURCE: Author 26/10/2018

The neurons take the input from **k** different neurons in the form of a column vector with **k** components and performs a certain process with this input to return a value. This process, $\boldsymbol{I(x)}$ is calculated as follows:

$$\boldsymbol{I(x)} = \vec{w}.\vec{x} + b$$

Where $\vec{\boldsymbol{w}}$ is the weights, $\vec{\boldsymbol{x}}$ is the input and $\boldsymbol{b}$ is the bias.

Each neuron passes on the calculated value of $I(x)$ to the next until it reaches the last one which needs to decide how to classify the given image.

The last neuron uses the function $H(\vec{w}.\vec{x} + b)$ to classify each image where

$$H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

The line $\overrightarrow{w}.\vec{x} + b = 0$ is called the decision boundary as it forms the barrier between an image classified as a dog and an image classified as something else. This can be plotted and used to classify images.

Calculating the weights and bias for a decision boundary

A loss function must be created to measure the errors from specific **weights** and **bias**.

Let x_i represent the ith image and y_i represent the outputs (dog or not a dog)

Given n images of our training data where $y_i = 1$ when the image is a dog and -1 when it is not a dog.

1) The **total number of incorrect classifications** is given by

$$\sum_{i=1}^{n} \left[[y_i \neq sign(w.x_i + b)] \right]$$

$\left[[x] \right]$ is an indicator function that outputs 1 if the statement inside is true and 0 if the statement inside is false. Whilst $sign(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0. \\ 1, & x > 0 \end{cases}$

So the function counts the number of times that the given output is wrong.

2) The distance of an image to the decision boundary is given by

$$Distance = \frac{w\ .x_i + b}{|w|}$$

The perception model starts with its weights, ⍰ equal to a vector of zeros and its bias $b = 0$. It then plots the images (in vector form) on a multidimensional plane and plots the decision boundary with the given weights and bias.

Each time the model encounters an incorrect classification it on an image

$(x_i,\ y_i)$ adjusts its weights and bias accordingly.

$$w = w_p + y_i x_i$$

$$b = b_p + y_i$$

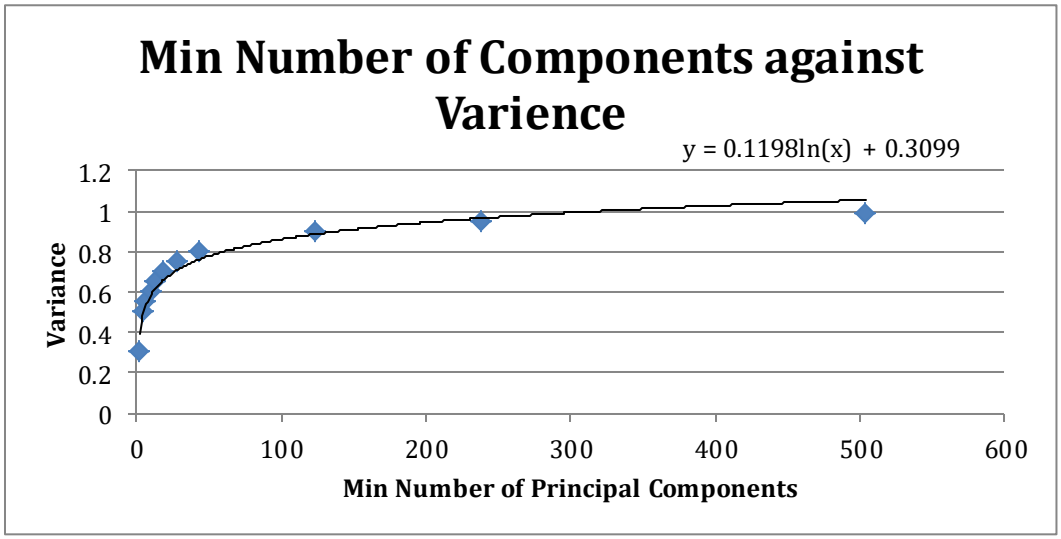
Where w_p and b_p are the previous weights and bias used. The perception model repeats this until there are no incorrect classifications.

Chapter 4: Results

4.1 Dimension Reduction Results

The results of the dimension reduction refer to the optimum number of principal components for Principal component analysis. Principal component analysis can plot a graph of the number of principal components and the total variance the components explain. In this paper, the minimum number of components that would produce different ratios of variance between the training data images were recorded. The results are:

Min Number of Components	Variance
2	0.3
5	0.5
7	0.55
10	0.6
13	0.65
19	0.7
29	0.75
44	0.8
124	0.9
238	0.95
504	0.99

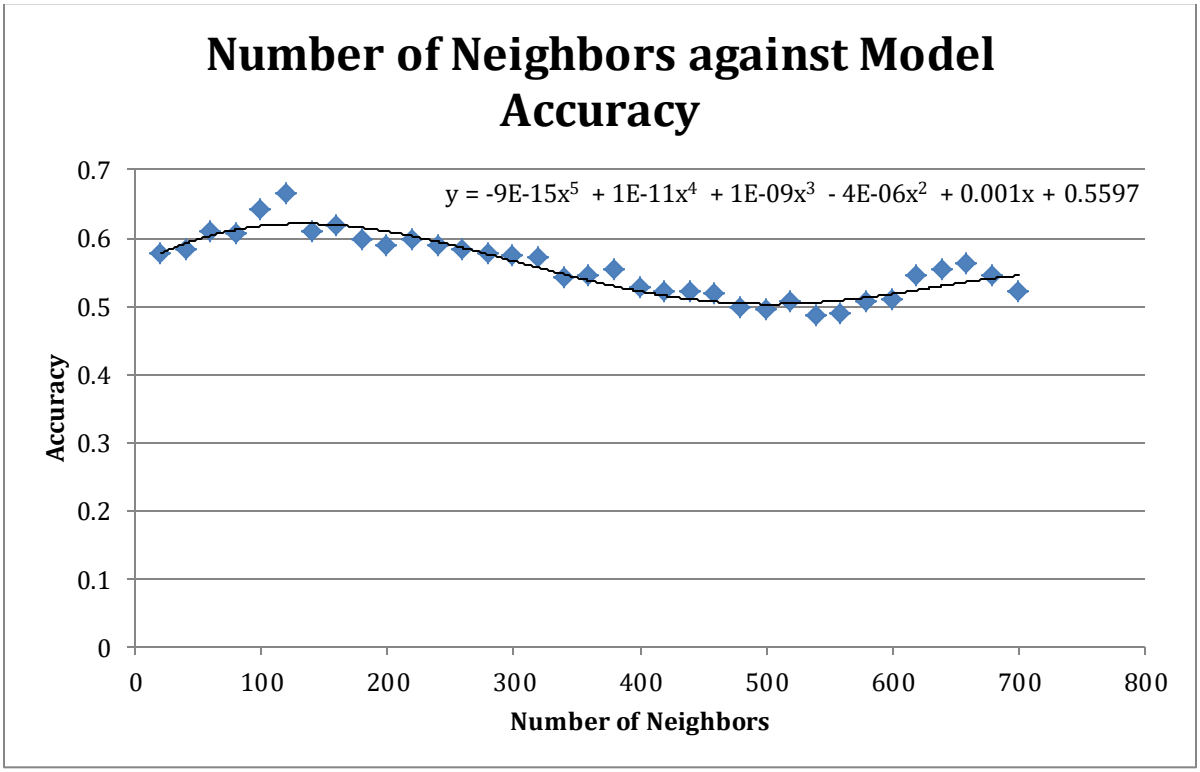


With the graph above one can see that the points are asymptotic to the line y = 1. Because of this there is no point in aiming for 100% variance. The table shows that 95% of the variance can be explained by 238 principal components this reduces the number of parameters needed by 94%.

4.2 Optimizing K-Nearest Neighbors

Neighbors	Accuracy
20	0.576
40	0.58
60	0.608
80	0.604
100	0.64
120	0.664
140	0.608
160	0.616
180	0.596
200	0.588
220	0.596
240	0.588
260	0.58
280	0.576
300	0.572
320	0.568
340	0.54
360	0.544
380	0.552
400	0.524
420	0.52
440	0.52
460	0.516
480	0.496
500	0.492
520	0.504
540	0.484
560	0.488
580	0.504
600	0.508
620	0.544
640	0.552
660	0.56
680	0.544
700	0.52

By plotting a graph of Number of Neighbors against Accuracy of Model, one can find a parametric equation to estimate the effect of the number of neighbors (x) on the accuracy of the k-NN model (y). Using this one can find the value of x that gives the maximum y value.



The graph gives the equation

$$y = (-9 \times 10^{-15})x^5 + (1 \times 10^{-11})x^4 + (1 \times 10^{-9})x^3 - (4 \times 10^{-6})x^2 + (0.001)x + 0.5597$$

However this equation has certain parameters

$$0 < x \leq 700$$

With this in mind one can find the local maximum rather than differentiating gives:

$$x = 145.9974 \;\; y = \; 0.6274948$$

Rounding up X one can expect to have the largest accuracy of **63%** when using **146** neighbors with k-NN.

4.3 Classification Models

A confusion matrix for binary classification (classifying into only two different classes) shows exactly where the model is making its mistakes and successes. It is in the form:

$$\begin{bmatrix} True\ Negatives & False\ Positives \\ False\ Negatives & True\ Positives \end{bmatrix}$$

True Negatives - The number of cats the model labelled as “cats”

False Positives – The number of dogs the model labelled as “cats”

False Negatives – The number of cats the model labelled as “dogs”

True Positives – The number of dogs the model labelled as “dogs”

Confusion matrices were used as a means of evaluating the model over just using accuracy like Turk and Pentland, because of the different characteristics that could be derived from the matrix. This allows for a more extensive evaluation of the model to best measure the effectiveness of a model, this is depicted below with k-Nearest Neighbors.

k-Nearest Neighbors Results

The k Nearest Neighbor confusion matrix for 146 neighbors:

$$\begin{bmatrix} 73 & 55 \\ 45 & 77 \end{bmatrix}$$

From the conclusion matrix one can get many results to help evaluate the model.

1) The accuracy of the model

$$\frac{True\ Positives + True\ Negatives}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives} = \frac{150}{250} = 60\%$$

2) The Error Rate

$$\frac{False\ Positives + False\ Negatives}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives} = \frac{100}{250}$$

$$= 40\%$$

3) True Positive Rate

$$\frac{True\ Positive}{True\ Positives + False\ Negatives} = \frac{77}{122} = 63\%$$

4) False Positive Rate

$$\frac{False\ Positive}{False\ Positives + True\ Negatives} = \frac{55}{128} = 43\%$$

5) True Negative Rate

$$\frac{True\ Negatives}{False\ Positives + True\ Negatives} = \frac{73}{128} = 67\%$$

6) Precision

$$\frac{True\ Positives}{False\ Positive + True\ Positives} = \frac{77}{132} = 58\%$$

(Markham)

Result Type	Percentage
Accuracy	60%
Error Rate	40%
True Positive Rate	63%
False Positive Rate	43%
True Negative Rate	67%
Precision	58%

Logistic Regression Results

Confusion Matrix: $\begin{bmatrix} 51 & 77 \\ 51 & 71 \end{bmatrix}$

Result Type	Percentage
Accuracy	49%
Error Rate	51%
True Positive Rate	68%
False Positive Rate	60%
True Negative Rate	40%
Precision	48%

Linear Progression Results

Confusion Matrix: $\begin{bmatrix} 18 & 89 \\ 18 & 125 \end{bmatrix}$

Result Type	Percentage
Accuracy	57%
Error Rate	43%
True Positive Rate	87%
False Positive Rate	83%
True Negative Rate	27%
Precision	58%

Chapter 5: Discussion of Results

5.1 Comparison

K Nearest Neighbors had the highest accuracy of 60% whilst Logistic Regression had an accuracy of 49% and Linear Perception had an accuracy of 57%. Logistic Regression's accuracy is below 50% so it can be perceived to be "guessing".

Linear Perception had the highest true positive rate (87%) whilst K Nearest Neighbors had the lowest (63%) , the true positive rate represents the percentage of the dogs that the model correctly identified. Since this model had the highest true positive rate, one can argue that it is the best model at identifying dogs and that K Nearest Neighbors is the worst.

However, Linear Perception had the highest false positive rate (83%), which is the percentage of images of cats that the model identified as "dogs". From this one can infer that although Linear Perception is very good at identifying dogs, it is bad a differentiating between dogs and cats. K Nearest Neighbors had the lowest false positive rate (43%), which is below 50% indicating that K Nearest Neighbors is the best at differentiating between cats and dogs.

The precision represents how often a model correctly identifies a dog when it predicts "dog". Both Linear Perception and k-Nearest Neighbors had the same precision score (58%), but if the precision score is specified then Linear Perception has a higher score by 0.01%.

It is fair to say that k-Nearest Neigbors with 146 neighbors is the best model for differentiating between images of dogs and cats due to its high accuracy & low false positive rate. However, Linear Perception is arguably the best model at identifying dogs due to its high true positive rate & its higher precision.

5.2 Limitations

All of the models were limited by the processing speed of the used laptop and the efficiency of the code. Both the processing speed & efficiency of the code limited the amount of images that the models could train with.

Principal Component Analysis

1) Picking too many principal components can result in producing unnecessary data which will hinder the classification models
2) Picking too little principal components can cause low accuracies and scores

k-Nearest Neighbors

1) Since k Nearest Neighbors relies on Euclidean distance to make its classifications, training data with large numbers of parameters would be harder to classify without the assistance of a dimension reducer.
2) K Nearest Neighbors finds the distance between all of the training data before making a classification. This can be time-consuming if the training data is large.
3) The optimum value of k can change with more training data.(Gillian)

Logistic Regression

1) Accuracy reduces when parameters are highly correlated (Ranganathan, and Aggarwal)
2) Too many parameters can cause inconstancies in training.

All of the classification models are limited to the data given by the dataset; this means that the model will only be as good as its dataset.

Chapter 6: Conclusion

6.1 Conclusion

The research question is: **"How effective are statistical models alongside dimension reduction at identifying images of dogs?"**

Statistical models are arguably very effective at identifying images of dogs, this is because they each provide different ways of identifying images of dogs with accuracies up to 60%. However, they are limited by the efficiency of the code and the processing speed of the used laptop. These limitations bring about unanswered questions such as "How does the size of the training set effect the accuracy of a model?" and "How does altering the parameters of models affect its accuracy?" these questions are not able to be answered due to the provided limitations.

Although statistical models cannot identify dogs with an accuracy of 100%, they are effective at identifying images of dogs to a large extent, despite its problem with differentiating between dogs and cats.

Chapter 7: References

Works Cited

Abdi, Hervé and L.J Williams. "Prinicpal Component Analysis." 23 August 2011. arXiv. 26 July 2018 <https://arxiv.org/pdf/1108.4372.pdf>.

Alonge, DO, and SA Abu. *Rabies In Ghana, youst Africa*. 1984, p. 1, https://www.ncbi.nlm.nih.gov/pubmed/6500862. Accessed 16 Oct 2018.

CodeEmporium. *Linear Regression And Multiple Regression.* 2017, https://www.youtube.com/watch?v=K_EH2abOp00&t=10s . Accessed 21 Aug 2018

Gockenbach, Mark. *"The Spectral Theorem For Symmetric Matrices"*. *Pages.Mtu.Edu.* 2012. http://pages.mtu.edu/~msgocken/ma5630spring2003/lectures/spectral/spectral/node2.html. Accessed 17 Aug 2018.

Jauregui, Jeff. *Principal Component Analysis With Linear Algebra*. Union College, 2018. pp, 4-9. http://www.math.union.edu/~jauregui/PCA.pdf. Accessed 17 Aug 2018.

Markham, Kevin. "Simple Guide To Confusion Matrix Terminology." *Data School*, 2018, https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/. Accessed 23 Aug 2018

Powell, Victor and Lewis Lehe. Principal Component Analysis Explained Visually. 2015. 26 July 2018 <http://setosa.io/ev/principal-component-analysis/>.

Ranganathan, Priva and Rakesh Aggarwal. "Common Pitfalls In Statistical Analysis: Linear Regression Analysis". *Perspectives In Clinical Research*, vol 8, no.2,2017, p. 100. *Medknow*, doi:10.4103/2229-3485,203040. Accessed 23 Aug 2018.

StatQuest, *Statquest; Principal Component Analysis(PCA), Step-By-Step*. 2018. https://www.youtube.com/watch?v=FgakZw6K1QQ. Accessed 8 July 2018.

Stroud, Kenneth A., and Dexter J. Booth. *Engineering Mathematics*. 7th ed., London, Palgrave Macmillan, 2013, pp. 509-10.

Zakka, Kevin. "A Complete Guide To K-Nearest-Neighbors With Applications In Pyrthon And R". *Kevin Zakka's Blog*, 2018, https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/. Accessed 20 Aug 2018