Conceptual: Chapter 4 — Classification

Import notebook functions

```
from notebookfuncs import *
```

Exercise 1

Using a little bit of algebra, prove that

$$p(X) = \frac{e^{\beta_0+\beta_1*X}}{1+e^{\beta_0+\beta_1*X}}$$

is equivalent to

$$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1*X}$$

.

In other words, the logistic function representation and logit representation for the logistic regression model are equivalent.

$$p(X) = \frac{e^{\beta_0+\beta_1*X}}{1+e^{\beta_0+\beta_1*X}}$$

$$\implies p(X) * (1+e^{\beta_0+\beta_1*X}) = e^{\beta_0+\beta_1*X}$$

$$\implies p(X) + p(X) * e^{\beta_0+\beta_1*X} = e^{\beta_0+\beta_1*X}$$

$$\implies p(X) = e^{\beta_0+\beta_1*X}(1-p(X))$$

$$\implies \frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1*X}$$

$$\implies \log\Big(\frac{p(X)}{1-p(X)}\Big) = \beta_0 + \beta_1 * X$$

Exercise 2

It was stated in the text that classifying an observation to the class for which

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu_k)^2)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu_l)^2)}$$

(4.17) is largest is equivalent to classifying an observation to the class for which

$$\delta_k(x) = x.\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k)$$

(4.18) is the largest. Prove that this is the case. In other words, under the assumption that the observations in the k_{th} class are drawn from a $N(\mu_k, \sigma^2)$ distribution, the Bayes classifier assigns an observation to the class for which the discriminant function is maximized.

By Bayes' theorem, we have

$$Pr\Big(Y = k|X = x\Big) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_x f_k(x)}$$

We change the notation to

$$p_k(x) = Pr\Big(Y = k|X = x\Big)$$

This is the posterior probability that an observation X = x belongs to the k_{th} class. That is, it is the probability that the observation belongs to the k_{th} class, given the predictor value for that observation.

Now, we assume that the observations in the k_{th} class are drawn from a $N(\mu_k, \sigma^2)$ distribution.

Therefore, substituting the normal distribution function for a Gaussian or normal distribution in the equation for $p_k(x)$, we have:

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu_k)^2)$$

and

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu_k)^2)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu_l)^2)}$$

We select the k_{th} classifier if the value of the above function is maximum amongst all the K classifiers, from 1 to K.

Since the denominator is a sum over all the K classifiers, it is constant. So the problem reduces to maximizing the value of the numerator.

$$\pi_k \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu_k)^2)$$

This is the same as maximizing the log of the above equation.

That is, we try to maximize:

$$log(\pi_k) - log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}(x-\mu_k)^2$$

$-log(\sqrt{2\pi}\sigma)$ is a constant as well. So that drops out as well.

Hence, we seek to maximize

$$log(\pi_k) - \frac{1}{2\sigma^2}(x-\mu_k)^2$$

Expanding the quadratic equation, we have

$$log(\pi_k) - \frac{1}{2\sigma^2}[x^2 + \mu_k^2 - 2x\mu_k]$$

Again, $x^2 * \frac{1}{2\sigma^2}$ is a constant in terms of k.

The equation reduces to

$$log(\pi_k) - \frac{1}{2\sigma^2}[\mu_k^2 - 2x\mu_k]$$

$$\log(\pi_k) - \mu_k^2/2\sigma^2 + \mu_k x/\sigma^2$$

Thus, the final equation becomes (rearranging the terms):

$$\delta_k = \frac{\mu_k}{\sigma^2}x - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

Exercise 3

This problem relates to the QDA model, in which the observations within each class are drawn from a normal distribution with a class specific mean vector and a class specific covariance matrix. We consider the simple case where p = 1; i.e. there is only one feature. Suppose that we have K classes, and that if an observation belongs to the kth class then X comes from a one-dimensional normal distribution, $X \sim N(\mu_k, \sigma_k^2)$. Recall that the density function for the one-dimensional normal distribution is given in

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} exp\Big(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2 \Big)$$

(4.16). Prove that in this case, the Bayes classifier is not linear. Argue that it is in fact quadratic.

Hint: For this problem, you should follow the arguments laid out in Section 4.4.1, but without making the assumption that $\sigma_1^2 = \cdots = \sigma_K^2$

By Bayes' theorem, we have

$$Pr\Big(Y = k|X = x\Big) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_x f_k(x)}$$

We change the notation to

$$p_k(x) = Pr\Big(Y = k|X = x\Big)$$

This is the posterior probability that an observation X = x belongs to the k_{th} class. That is, it is the probability that the observation belongs to the k_{th} class, given the predictor value for that observation.

Now, we assume that the observations in the k_{th} class are drawn from a $N(\mu_k, \sigma_k^2)$ distribution.

Therefore, substituting the normal distribution function for a Gaussian or normal distribution in the equation for $p_k(x)$, we have:

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2)$$

and

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma_l} exp(-\frac{1}{2\sigma_l^2}(x-\mu_l)^2)}$$

We select the k_{th} classifier if the value of the above function is maximum amongst all the K classifiers, from 1 to K.

Since the denominator is a sum over all the K classifiers, it is constant. So the problem reduces to maximizing the value of the numerator.

$$\pi_k \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2)$$

This is the same as maximizing the log of the above equation and removing the constant term $1/\sqrt{2\pi}$

That is, we try to maximize:

$$log(\pi_k) - log(\sigma_k) - \frac{1}{2\sigma_k^2}(x-\mu_k)^2$$

$$log(\pi_k) - log(\sigma_k) - \frac{1}{2\sigma_k^2}[x^2 - 2\mu_k x + \mu_k^2]$$

$$-\frac{x^2}{2\sigma_k^2} + \frac{\mu_k x}{\sigma_k^2} - \frac{\mu_k^2}{2\sigma_k^2} - log(\sigma_k) + log(\pi_k)$$

$$\delta_k = -\frac{x^2}{2\sigma_k^2} + \frac{\mu_k x}{\sigma_k^2} - \frac{\mu_k^2}{2\sigma_k^2} - log(\sigma_k) + log(\pi_k)$$

The above function is quadratic in x with non-zero coefficients for both x and x^2. Thus, the Bayes classifier is not linear, but quadratic.

Exercises 3 and 4

Alternatively, we can derive the δ form of the LDA and QDA as follows:

$$P(y=k|x) = \frac{P(x|y=k)P(y=k)}{P(x)} = \frac{P(x|y=k)P(y=k)}{\sum_l P(x|y=l)P(y=l)}$$

And we select the class that maximizes this posterior probability.

When P(x|y) is modeleed as a multivariate Gaussian distribution with density:

$$P(x|y=k) = p_k(x) = \frac{1}{(2\pi)^{d/2}|\ \Sigma_k\ |^{1/2}} exp\Big(-\frac{1}{2}(x-\mu_k)^T {\Sigma_k}^{-1}(x-\mu_k)\Big)$$

wherre d is the number of features.

According to the model above, the log of the posterior is:

$$log(P(x|y=k)) = log(P(x|y=k)) + log(P(y=k)) + C_{st}$$

where the constant term C_{st} corresponds to the denominator P(x) , in addition to other constant terms from the Gaussian. The predicted class is the one that maximises this log-posterior.

$$= -\frac{1}{2} log \mid \Sigma_k \mid -\frac{1}{2}(x-\mu_k)^T {\Sigma_k}^{-1}(x-\mu_k) + log(P(y=k)) + C_{st}$$

$$= -\frac{1}{2} log \mid \Sigma_k \mid -\frac{1}{2}[x^T\Sigma_k^{-1}x - \mu_k^T\Sigma_k^{-1}x - x^T\Sigma_k^{-1}\mu_k + \mu_k^T\Sigma_k^{-1}\mu_k] + log(\pi_k) + C_{st}$$

$$= -\frac{1}{2} log \mid \Sigma_k \mid -\frac{1}{2}[x^T\Sigma_k^{-1}x - 2x^T\Sigma_k^{-1}\mu_k + \mu_k^T\Sigma_k^{-1}\mu_k] + log(\pi_k) + C_{st}$$

Hence for QDA, we have the following delta equation to be maximized to select the classifier k with the maximum probability.

$$\delta_k = -\frac{1}{2} log \mid \Sigma_k \mid -\frac{1}{2}[x^T\Sigma_k^{-1}x - 2x^T\Sigma_k^{-1}\mu_k + \mu_k^T\Sigma_k^{-1}\mu_k] + log(\pi_k) + C_{st}$$

$$\delta_k = -\frac{x^T\Sigma_k^{-1}x}{2} + x^T\Sigma_k^{-1}\mu_k + \frac{\mu_k^T\Sigma_k^{-1}\mu_k}{2} + log(\pi_k) - \frac{1}{2} log \mid \Sigma_k \mid + C_{st}$$

This is quadratic in x.

In the case of LDA, we have a common covariance matrix, Σ instead of Σ_k for each classifier class.

$$\delta_k = -\frac{x^T\Sigma^{-1}x}{2} + x^T\Sigma^{-1}\mu_k + \frac{\mu_k^T\Sigma^{-1}\mu_k}{2} + log(\pi_k) - \frac{1}{2} log \mid \Sigma \mid + C_{st}$$

Here, $x^T\Sigma_k^{-1}x$ and $log \mid \Sigma \mid$ are constant in k. Hence, these can be merged into a constant term, C_{stadj}

$$\delta_k = x^T\Sigma^{-1}\mu_k + \frac{\mu_k^T\Sigma^{-1}\mu_k}{2} + log(\pi_k) + C_{stadj}$$

This equation is linear in x.

Exercise 4

When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that non-parametric approaches often perform poorly when p is large. We will now investigate this curse.

(a)

Suppose that we have a set of observations, each with measurements on p = 1 feature, X. We assume that X is uniformly (evenly) distributed on [0, 1]. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10 % of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with X = 0.6, we will use observations in the range [0.55, 0.65]. On average, what fraction of the available observations will we use to make the prediction?

X is uniformly distributed on [0, 1]. Therefore, X follows the standard uniform distribution where each point is equally likely. We wish to predict a test observation's response using only observations that are within 10% of the range of X closest to that test observation. This implies that for observations in the range [0.5, 0.95] , we use 10% of the closest observations on either side of the test observation. That is, for 90% of the observations in the range [0,1], we use 10% of the closest observations, 5% on either side of the test observation. For observations that fall in the range [0, 0.5] , we are restricted to observations in the range [0, 0.5], [0, 0.6], [0, 0.7] , [0,0.8], [0,0.9] and [0.0.10], which would make it 5, 6, 7, 8, 9 and 10 percentage points respectively with an average of

$$\frac{5+6+7+8+9+10}{6} = \frac{45}{6} = 7.5\%$$

Thus, on average, we would use

$$0.9 * 10 + 0.1 * 7.5 = 9 + 0.75 = 9.75\% \, of \, the \, observations.$$

Explanation: The question states we are using only observations that are within 10 % of the range of X closest to that test observation and not 10% (or rather k) of the closest observations like we would do under KNN.

(b)

Now suppose that we have a set of observations, each with measurements on p = 2 features, X_1 and X_2 . We assume that (X_1, X_2) are uniformly distributed on [0, 1] $\times$ [0, 1]. We wish to predict a test observation's response using only observations that are within 10 % of the range of X_1 and within 10 % of the range of X_2 closest to that test observation. For instance, in order to predict the response for a test observation with X_1 = 0.6 and X_2 = 0.35, we will use observations in the range [0.55, 0.65] for X_1 and in the range [0.3, 0.4] for X_2 . On average, what fraction of the available observations will we use to make the prediction?

We have 9.75% of observations that satisfy the range [max(X1-0.05,0), min(X1+0.05, 1)] where X1 is the first predictor. Similarly, We have 9.75% of observations that satisfy the range [max(X2-0.05,0), min(X2+0.05, 1)] where X2 is the second predictor. For the observations to satisfy both criteria and assuming that X1 and X2 are independent of each other, the fraction of available observations used to make the predictions are:

```
print(f"{(0.0975 * 0.0975) * 100}%")
```

```
0.950625%
```

(c)

Now suppose that we have a set of observations on p = 100 features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10% of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?

```
print(f"{(0.0975 ** 100) * 100}% which is close to zero.")
```

```
7.951728986183188e-100% which is close to zero.
```

(d)

Using your answers to parts (a)–(c), argue that a drawback of KNN when p is large is that there are very few training observations "near" any given test observation.

Since the $\lim_{p\to\infty}(0.0975)^p$ approaches zero as p increases, it is quite likely that there will be very few or no points actually "near" any given test observation. Thus, a drawback of KNN is that the K nearest neighbors chosen won't be very close at all and thus may not be the best estimators for the given test observation.

(e)

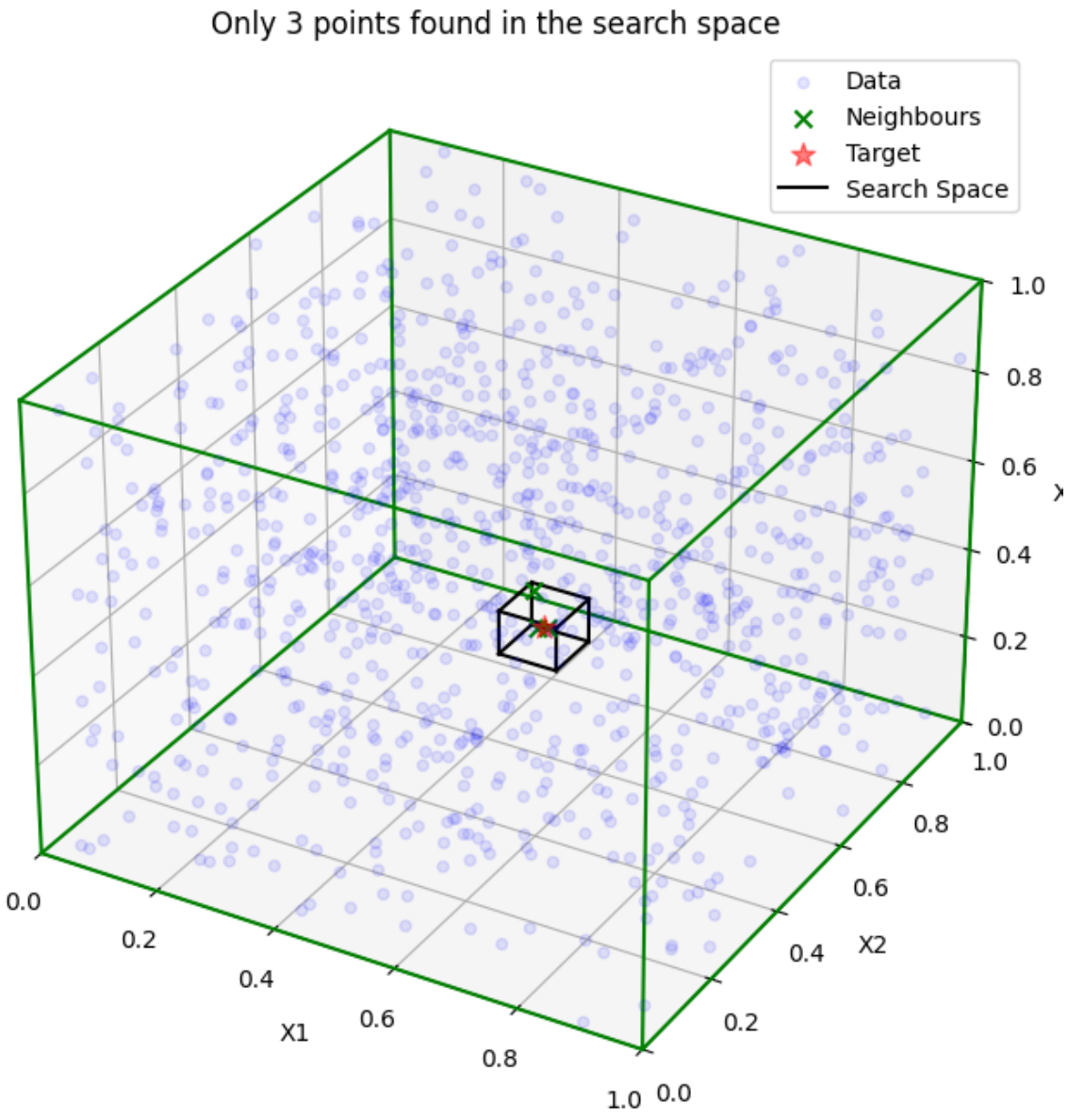
Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube segment, centered around the test observation that contains, on average, 10 % of the training observations. For p = 1, 2, and 100, what is the length of each side of the hypercube? Comment on your answer.

Note: A hypercube is a generalization of a cube to an arbitrary number of dimensions. When $p = 1$, a hypercube is simply a line segment, when $p = 2$ it is a square, and when $p = 100$ it is a 100-dimensional cube.

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

rng = np.random.RandomState(0)

# Generate random x, y, z coordinates from uniform distribution
x = rng.uniform(0, 1, 1000)
y = rng.uniform(0, 1, 1000)
z = rng.uniform(0, 1, 1000)

# Target point
target_x, target_y, target_z = 0.5, 0.6, 0.3

# Filter radius (5% on each side)
filter_radius = 0.0975 / 2

# Calculate filter bounds
filter_x_min = max(target_x - filter_radius, 0)
filter_x_max = min(target_x + filter_radius, 1)
filter_y_min = max(target_y - filter_radius, 0)
filter_y_max = min(target_y + filter_radius, 1)
filter_z_min = max(target_z - filter_radius, 0)
filter_z_max = min(target_z + filter_radius, 1)

# Filter points
```

```
mask = (filter_x_min <= x) & (x <= filter_x_max) & \
       (filter_y_min <= y) & (y <= filter_y_max) & \
       (filter_z_min <= z) & (z <= filter_z_max)

# Create the 3D plot
fig = plt.figure(figsize=(8,14))
ax = fig.add_subplot(111, projection='3d')

# Plot points outside filter
ax.scatter(x[~mask], y[~mask], z[~mask], c='b', alpha=0.1, label="Data")

# Plot points within filter
ax.scatter(x[mask], y[mask], z[mask], c='g', s=50, marker='x', alpha=1.0, label="Neighbours")

ax.set_title(f"Only {len(x[mask])} points found in the search space")

# Plot target point
ax.scatter(target_x, target_y, target_z, c='r', s=100, marker='*', label="Target",alpha=0.5)

# Plot larger cube around boundaries
cube_vertices = np.array([
    [0, 0, 0],
    [1, 0, 0],
    [1, 1, 0],
    [0, 1, 0],
    [0, 0, 1],
    [1, 0, 1],
    [1, 1, 1],
    [0, 1, 1]
])

cube_edges = np.array([
    [0, 1],
    [1, 2],
    [2, 3],
    [3, 0],
    [4, 5],
    [5, 6],
    [6, 7],
    [7, 4],
    [0, 4],
    [1, 5],
```

```
    [2, 6],
    [3, 7]
])

for edge in cube_edges:
    ax.plot3D(*zip(cube_vertices[edge[0]], cube_vertices[edge[1]]), c='g')

# Plot smaller cube around filtered points
cube_vertices = np.array([
    [filter_x_min, filter_y_min, filter_z_min],
    [filter_x_max, filter_y_min, filter_z_min],
    [filter_x_max, filter_y_max, filter_z_min],
    [filter_x_min, filter_y_max, filter_z_min],
    [filter_x_min, filter_y_min, filter_z_max],
    [filter_x_max, filter_y_min, filter_z_max],
    [filter_x_max, filter_y_max, filter_z_max],
    [filter_x_min, filter_y_max, filter_z_max]
])

for edge in cube_edges:
    ax.plot3D(*zip(cube_vertices[edge[0]], cube_vertices[edge[1]]), c='k')

ax.plot3D(0,0,0,c='k', label="Search Space")

# Set plot limits
ax.set_xlim(0, 1)
ax.set_ylim(0, 1)
ax.set_zlim(0, 1)

# Set labels
ax.set_xlabel('X1')
ax.set_ylabel('X2')
ax.set_zlabel('X3')

# Show grid
ax.grid(True)
ax.legend()
plt.show()
```



In high dimensional space, points that are drawn from a distribution tend to be far away from each other.

Let us consider uniform distribution over a hypercube $[0,1]^d$

Q: What is the probability that is inside the small cube?

A: Volume(small cube)/volume($[0,1]^d$) $= l^d$

Now assume we sample points uniform randomly, and we observe K points fall inside the small cube. So empirically, the probability of sampling a point inside the small cube is roughly $\frac{K}{n}$. Thus, we have $l^d \approx \frac{K}{n}$.

Q: How large we should set l s.t., we will have K examples (out of n) fall inside the small cube?

A: $l \approx (\frac{K}{n})^{1/d} \to 1$ as $d \to \infty$

Bad news: When $d \to \infty$, the K nearest neighbors will be all over the place! (Cannot trust them, as they are not nearby points anymore!)

Reference: 1. https://www.cs.cornell.edu/courses/cs4780/2022fa/slides/KNN_annotated.pdf

Here, $\frac{K}{n} = 0.0975$ as calculated earlier.

```
no_of_parameters = np.array([1,2,100])
l = [(0.0975) ** (1/p) for p in no_of_parameters]
```

```
[0.0975, 0.3122498999199199, 0.9769898372300478]
```

Exercise 5

We now examine the differences between LDA and QDA.

(a)

If the Bayes decision boundary is linear, do we expect LDA or QDA to perform better on the training set? On the test set?

If the Bayes decision boundary is linear, QDA may perform better on the training set since it may tend to overfit the data. However, on the test set, it is LDA that will fit better to outside data not seen or encountered in the training set.

(b)

If the Bayes decision boundary is non-linear, do we expect LDA or QDA to perform better on the training set? On the test set?

If the Bayes decision boundary is non-linear, QDA will perform better on the training set since it will tend to overfit the data However, on the test set, QDA will definitely fit better to outside data not seen or encountered in the training set.

(c)

In general, as the sample size n increases, do we expect the test prediction accuracy of QDA relative to LDA to improve, decline, or be unchanged? Why?

As the sample size increases, the variance of the classifier is not a major concern or when obviously the assumption of a common covariance for the K classes is untenable. The QDA test prediction accuracy will increase when the training set is very large.

(d)

True or False: Even if the Bayes decision boundary for a given problem is linear, we will probably achieve a superior test error rate using QDA rather than LDA because QDA is flexible enough to model a linear decision boundary. Justify your answer.

False. The QDA may overfit on the training data when the Bayes decision boundary is linear but the variance will increase when exposed to out-of-sample test data. This will increase the test error rate where LDA will prove superior since it actually matches the linear decision boundary more closely.

Exercise 6

Suppose we collect data for a group of students in a statistics class with variables $X_1 = hours\ studied, X_2 = undergrad\ GPA$, and $Y = receive\ an\ A$. We fit a logistic regression and produce estimated coefficient, $\beta_0 = -6, \beta_1 = 0.05, \beta_2 = 1$.

```
from sympy import Symbol
from mpmath import e
from algebra_with_sympy import Eqn, solve

X1 = Symbol("hours.studied")
X2 = Symbol("undergrad.GPA")
Y = Symbol("receive.an.A")
beta_0 = Symbol(" ")
beta_1 = Symbol(" ")
beta_2 = Symbol(" ")

exponent = beta_0 + beta_1 * X1 + beta_2 * X2
numerator = e ** exponent
denominator = 1 + numerator
```

```
logit_probability = numerator / denominator
logit_probability = logit_probability.subs([(beta_0, -6),(beta_1,0.05),(beta_2,1)])
```

$$\frac{2.71828182845905^{0.05hours.studied+undergrad.GPA-6}}{2.71828182845905^{0.05hours.studied+undergrad.GPA-6}+1}$$

(a)

Estimate the probability that a student who studies for 40 hours and has an undergrad GPA of 3.5 gets an A in the class.

```
hours_studied = 40
undergrad_GPA = 3.5
logit_probability.subs([(X1, hours_studied),(X2, undergrad_GPA)])
```

0.377540668798145

(b)

How many hours would the student in part (a) need to study to have a 50% chance of getting an A in the class?

```
prob = logit_probability.subs(X2, undergrad_GPA)
eqn_to_solve = Eqn(0.5, prob)
solve(eqn_to_solve,X1)
```

$\{hours.studied = 50.0\}$

Exercise 7

Suppose that we wish to predict whether a given stock will issue a dividend this year ("Yes" or "No") based on X, last year's percent profit. We examine a large number of companies and discover that the mean value of X for companies that issued a dividend was $\overline{X} = 10$, while the mean for those that didn't was $\overline{X} = 0$. In addition, the variance of X for these two sets of companies was $\sigma^2 = 36$. Finally, 80% of companies issued dividends. Assuming that X follows a normal distribution, predict the probability that a company will issue a dividend this year given that its percentage profit was X = 4 last year.

Hint: Recall that the density function for a normal random variable is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-)^2/2\sigma^2}$$

. You will need to use Bayes' theorem.

By Bayes' theorem, we have

$$P(Y = k|X = x) = \frac{\pi_k * f_k(x)}{\sum_l \pi_l * f_l(x)}$$

Now the relevant terms from the above problem statement are as follows:

X - Last year's profit percentage for the company

$\bar{X}_y$ - The mean profit percent for companies that declared dividends = 10

$\bar{X}_n$ - The mean profit percent for companies that did not declare dividends = 0

σ^2 - The variance of X for these two sets of companies = 36

$\implies \sigma = \sqrt{36} = 6$

π_y - Percentage or proportion of companies that declared dividends = 0.8

π_n - Percentage or proportion of companies that did not declare dividends = 0.2

Predict the probability that a company will issue a dividend this year given that its percentage profit was X = 4 last year.

```
from sympy import sqrt, Integer, pi
sigma = Symbol(" ")
x = Symbol("x")
mu_yes = Symbol(" ")
mu_no = Symbol(" ")
x_bar_yes = Symbol("x̄ ")
x_bar_no = Symbol("x̄ ")
pi_yes = Symbol(" ")
pi_no = Symbol(" ")
density_fn_const = 1 /  ((2 * pi)**(1/2) * sigma)
density_fn_exponent_yes = ((x - mu_yes) ** 2) / (2 * sigma ** 2)
density_fn_exponent_no = ((x - mu_no) ** 2) / (2 * sigma ** 2)
density_fn_yes = (density_fn_const) * e ** -(density_fn_exponent_yes)
probability_yes = pi_yes * density_fn_yes
density_fn_no = (density_fn_const) * e ** -(density_fn_exponent_no)
probability_no = pi_no * density_fn_no
probability_x = probability_yes + probability_no
```

$$\frac{\sqrt{2}2.71828182845905^{-\frac{(x-\mu_n)^2}{2\sigma^2}}\pi_n}{2\sqrt{\pi}\sigma} + \frac{\sqrt{2}2.71828182845905^{-\frac{(x-\mu_\gamma)^2}{2\sigma^2}}\pi_\gamma}{2\sqrt{\pi}\sigma}$$

The probability that a company will issue a dividend this year given that its percentage profit was X = 4 last year is as follows:

```
conditional_prob = probability_yes / probability_x
conditional_prob.subs([(mu_no, 0), (mu_yes, 10), (sigma,6), (pi_yes, 0.8), (pi_no,0.2), (x,4)
```

0.751852453297526

Exercise 8

Suppose that we take a data set, divide it into equally-sized training and test sets, and then try out two different classification procedures. First we use logistic regression and get an error rate of 20% on the training data and 30% on the test data. Next we use 1-nearest neighbors (i.e. K = 1) and get an average error rate (averaged over both test and training data sets) of 18%. Based on these results, which method should we prefer to use for classification of new observations? Why?

Logistic Regression Training Error Rate = 20%. Logistic Regression Testing Error Rate = 30%. 1-NN Average Error Rate = 18%.

The 1-NN Average Error Rate seems better at first appearances. However, it's misleading since 1-NN training error rate is always zero. Thus, the test error rate for 1-NN is double the average error rate = 2 * 18% = 36%.

Thus, we would prefer Logistic Regression for classification of new observations over 1-NN in this case since its test error rate is lower by 6 percentage points.

Exercise 9

This problem has to do with odds.

Odds are defined as $\frac{P(X)}{1-P(X)}$.

(a)

On average, what fraction of people with an odds of 0.37 of defaulting on their credit card payment will in fact default?

This implies that $\frac{P(X)}{1-P(X)} = 0.37$

```
odds = 0.37
probability = Symbol("P(X)")
odds_formula = probability / (1 - probability)
print(odds_formula)
odds_equation = Eqn(odds, odds_formula )
print(odds_equation)
solve(odds_equation, probability)
```

```
P(X)/(1 - P(X))
0.370000000000000 = P(X)/(1 - P(X))
```

$\{P(X) = 0.27007299270073\}$

(b)

Suppose that an individual has a 16% chance of defaulting on her credit card payment. What are the odds that she will default?

```
prob_value = 0.16
odds_formula.subs(probability, prob_value)
```

0.19047619047619

Exercise 10

Equation 4.32 derived an expression for $log\left(\frac{Pr(Y=k|X=x)}{Pr(Y=K|X=x)}\right)$ **in the setting where p > 1, so that the mean for the** k_{th} **class,** μ_k **, is a p-dimensional vector, and the shared covariance** $\sum$ **is a p** $\times$ **p matrix. However, in the setting with p = 1, (4.32) takes a simpler form, since the means** $\mu_1, ..., \mu_K$ **and the variance** σ^2 **are scalars. In this simpler setting, repeat the calculation in (4.32), and provide expressions for** a_k **and** b_{kj} **in terms of** π_k **,** π_K **,** μ_k **,** μ_K **, and** σ^2**.**

Equation 4.32:

$$log\bigg(\frac{Pr(Y=k|X=x)}{Pr(Y=K|X=x)}\bigg) = a_k + \sum_{j=1}^{p} b_{kj}x_j$$

In the simpler form of equation 4.32 with p =1 , $log\bigg(\frac{Pr(Y=k|X=x)}{Pr(Y=K|X=x)}\bigg)$ becomes

$$log\bigg(\frac{Pr(Y=k|X=x)}{Pr(Y=K|X=x)}\bigg) = log\bigg(\frac{\pi_k f_k(x)}{\pi_K f_K(x)}\bigg)$$

$$= log\bigg(\frac{\pi_k \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{1}{2}(x-\mu_k)^2/\sigma^2)}{\pi_K \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{1}{2}(x-\mu_K)^2/\sigma^2)}\bigg)$$

$$= log\bigg(\frac{\pi_k}{\pi_K}\bigg) - \frac{1}{2}\frac{(x-\mu_k)^2}{\sigma^2} + \frac{1}{2}\frac{(x-\mu_K)^2}{\sigma^2}$$

$$= log\bigg(\frac{\pi_k}{\pi_K}\bigg) - \frac{1}{2}\frac{x^2}{\sigma^2} + \frac{1}{2}\frac{x^2}{\sigma^2} + \frac{x\mu_k}{\sigma^2} - \frac{x\mu_K}{\sigma^2} - \frac{1}{2}\frac{\mu_k^2}{\sigma^2} + \frac{1}{2}\frac{\mu_K^2}{\sigma^2}$$

$$= log\bigg(\frac{\pi_k}{\pi_K}\bigg) - \frac{1}{2}\frac{\mu_k^2}{\sigma^2} + \frac{1}{2}\frac{\mu_K^2}{\sigma^2} + \frac{x\mu_k}{\sigma^2} - \frac{x\mu_K}{\sigma^2}$$

$$= log\bigg(\frac{\pi_k}{\pi_K}\bigg) - \frac{1}{2}\frac{\mu_k^2 - \mu_K^2}{\sigma^2} + (\frac{\mu_k - \mu_K}{\sigma^2})x$$

Here, we have the solution

$$a_k = log\bigg(\frac{\pi_k}{\pi_K}\bigg) - \frac{1}{2}\frac{(\mu_k - \mu_K)(\mu_k + \mu_K)}{\sigma^2}$$

and

$$b_k = (\frac{\mu_k - \mu_K}{\sigma^2})$$

and the expanded equation is linear in x.

Exercise 11

Work out the detailed forms of a_k , b_{kj} , and c_{kjl} in

$$log\left(\frac{Pr(Y=k|X=x)}{Pr(Y=K|X=x)}\right) = a_k + \sum_{j=1}^{p} b_{kj}x_j + \sum_{j=1}^{p}\sum_{l=1}^{p} c_{kjl}x_j x_l$$

(4.33). Your answer should involve π_k , π_K , μ_k , μ_K , $\sum_k$, and $\sum_K$.

$$log\left(\frac{Pr(Y=k|X=x)}{Pr(Y=K|X=x)}\right) = log\left(\frac{\pi_k f_k(x)}{\pi_K f_K(x)}\right)$$

$$= log\left(\frac{\pi_k exp(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k))}{\pi_K exp(-\frac{1}{2}(x-\mu_K)^T\Sigma_K^{-1}(x-\mu_K))}\right)$$

$$= log(\frac{\pi_k}{\pi_K}) - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \frac{1}{2}(x-\mu_K)^T\Sigma_K^{-1}(x-\mu_K)$$

$$= log\left(\frac{\pi_k}{\pi_K}\right) - \frac{1}{2}\left[x^T\Sigma_k^{-1}x - x^T\Sigma_k^{-1}\mu_k - \mu_k^T\Sigma_k^{-1}x + \mu_k^T\Sigma_k^{-1}\mu_k\right] + \frac{1}{2}\left[x^T\Sigma_K^{-1}x - x^T\Sigma_K^{-1}\mu_K - \mu_K^T\Sigma_K^{-1}x + \mu_K^T\Sigma_K^{-1}\mu_K\right]$$

$$\because A^T x B = B^T x A$$
$$x^T\Sigma_k^{-1}\mu_k = \mu_k^T\Sigma_k^{-1}x \text{ and}$$
$$x^T\Sigma_K^{-1}\mu_K = \mu_K^T\Sigma_K^{-1}x$$
$$\Sigma_k \text{ and } \Sigma_K \text{ are symmetric}$$
$$\text{covariance matrices and}$$
$$\text{their inverses are also symmetric.}$$

$$\therefore \text{we have}$$
$$log\left(\frac{Pr(Y=k|X=x)}{Pr(Y=K|X=x)}\right)$$

$$= log\left(\frac{\pi_k}{\pi_K}\right) - \frac{1}{2}\left[x^T\Sigma_k^{-1}x - 2x^T\Sigma_k^{-1}\mu_k + \mu_k^T\Sigma_k^{-1}\mu_k\right] + \frac{1}{2}\left[x^T\Sigma_K^{-1}x - 2x^T\Sigma_K^{-1}\mu_K + \mu_K^T\Sigma_K^{-1}\mu_K\right]$$

$$= log\left(\frac{\pi_k}{\pi_K}\right) + \frac{1}{2}\left(\mu_K^T\Sigma_K^{-1}\mu_K - \mu_k^T\Sigma_k^{-1}\mu_k\right) - \left(x^T\Sigma_K^{-1}\mu_K - x^T\Sigma_k^{-1}\mu_k\right) + \frac{1}{2}\left(x^T\Sigma_K^{-1}x - x^T\Sigma_k^{-1}x\right)$$

$$= log\left(\frac{\pi_k}{\pi_K}\right) + \frac{1}{2}\left(\mu_K^T\Sigma_K^{-1}\mu_K - \mu_k^T\Sigma_k^{-1}\mu_k\right) - x^T(\Sigma_K^{-1}\mu_K - \Sigma_k^{-1}\mu_k) + \frac{1}{2}x^T(\Sigma_K^{-1} - \Sigma_k^{-1})x$$

Here, we have

$$a_k = log\Big(\frac{\pi_k}{\pi_K}\Big) + \frac{1}{2}\Big(\mu_K^T \Sigma_K^{-1} \mu_K - \mu_k^T \Sigma_k^{-1} \mu_k\Big)$$
$$b_{kj}\ is\ the\ j_{th}\ component\ of\ \Sigma_K^{-1}\mu_K - \Sigma_k^{-1}\mu_k$$
$$c_{kjl}\ is\ the\ jl_{th}\ component\ of\ \Sigma_K^{-1} - \Sigma_k^{-1}$$

And the QDA equation becomes:

$$log\Big(\frac{Pr(Y=k|X=x)}{Pr(Y=K|X=x)}\Big)$$
$$= a_k + \sum_{j=1}^{p} b_{kj}x_j + \sum_{j=1}^{p}\sum_{l=1}^{p} x_j c_{kjl} x_l$$

- LDA is a special case of QDA with $c_{jkl} = 0$ for all j = 1, . . . , p, l = 1, . . . , p, and k = 1, . . . , K. (Of course, this is not surprising, since LDA is simply a restricted version of QDA with $\Sigma_1 = \cdot \cdot \cdot = \Sigma_K = \Sigma$.)
- This can be seen in the above equation as well where $\Sigma_K^{-1} - \Sigma_k^{-1} = 0$ when the covariance matrix is common as in for LDA.

Exercise 12

Suppose that you wish to classify an observation X R into apples and oranges. You fit a logistic regression model and find that

$$\hat{Pr}(Y = orange|X = x) = \frac{exp(\hat{\beta}_0 + \hat{\beta}_1 x)}{1 + exp(\hat{\beta}_0 + \hat{\beta}_1 x)}$$

.

Your friend fits a logistic regression model to the same data using the softmax formulation in

$$\hat{Pr}(Y = orange|X = x) = \frac{e^{\beta_0 + \beta_1 x_1 + ... + \beta_p x_p}}{\sum_{l=1}^{K} e^{\beta_0 + \beta_1 x_1 + ... + \beta_p x_p}}$$

(4.13), and finds that

$$\hat{Pr}(Y = orange|X = x) = \frac{exp(\hat{\alpha}_{orange0} + \hat{\alpha}_{orange1} x)}{exp(\hat{\alpha}_{orange0} + \hat{\alpha}_{orange1} x) + exp(\hat{\alpha}_{apple0} + \hat{\alpha}_{apple1} x)}$$

(a)

What is the log odds of orange versus apple in your model?

$$\begin{aligned}
\hat{Pr}(Y = orange|X = x) &= \frac{exp(\hat{\beta}_0 + \hat{\beta}_1 x)}{1 + exp(\hat{\beta}_0 + \hat{\beta}_1 x)} \\
\implies \hat{Pr}(Y = apple|X = x) &= 1 - (\frac{exp(\hat{\beta}_0 + \hat{\beta}_1 x)}{1 + exp(\hat{\beta}_0 + \hat{\beta}_1 x)}) \\
&= \frac{1 + exp(\hat{\beta}_0 + \hat{\beta}_1 x) - exp(\hat{\beta}_0 + \hat{\beta}_1 x)}{1 + exp(\hat{\beta}_0 + \hat{\beta}_1 x)} \\
&= \frac{1}{1 + exp(\hat{\beta}_0 + \hat{\beta}_1 x)} \\
\implies \frac{\hat{Pr}(Y = orange|X = x)}{\hat{Pr}(Y = apple|X = x)} &= exp(\hat{\beta}_0 + \hat{\beta}_1 x) \\
\implies log\Big(\frac{\hat{Pr}(Y = orange|X = x)}{\hat{Pr}(Y = apple|X = x)}\Big) &= \hat{\beta}_0 + \hat{\beta}_1 x
\end{aligned}$$

(b)

What is the log odds of orange versus apple in your friend's model?

$$\begin{aligned}
\hat{Pr}(Y = orange|X = x) &= \frac{exp(\hat{\alpha}_{orange0} + \hat{\alpha}_{orange1} x)}{exp(\hat{\alpha}_{orange0} + \hat{\alpha}_{orange1} x) + exp(\hat{\alpha}_{apple0} + \hat{\alpha}_{apple1} x)} \\
\implies \hat{Pr}(Y = apple|X = x) &= \frac{exp(\hat{\alpha}_{apple0} + \hat{\alpha}_{apple1} x)}{exp(\hat{\alpha}_{orange0} + \hat{\alpha}_{orange1} x) + exp(\hat{\alpha}_{apple0} + \hat{\alpha}_{apple1} x)} \\
\implies \frac{\hat{Pr}(Y = orange|X = x)}{\hat{Pr}(Y = apple|X = x)} &= \frac{exp(\hat{\alpha}_{orange0} + \hat{\alpha}_{orange1} x)}{exp(\hat{\alpha}_{apple0} + \hat{\alpha}_{apple1} x)} \\
\implies log\Big(\frac{\hat{Pr}(Y = orange|X = x)}{\hat{Pr}(Y = apple|X = x)}\Big) &= \hat{\alpha}_{orange0} + \hat{\alpha}_{orange1} x - \hat{\alpha}_{apple0} - \hat{\alpha}_{apple1} x \\
&= \hat{\alpha}_{orange0} - \hat{\alpha}_{apple0} + (\hat{\alpha}_{orange1} - \hat{\alpha}_{apple1}) x
\end{aligned}$$

(c)

Suppose that in your model, $\beta_0 = 2$ and $\beta_1 = -1$. What are the coefficient estimates in your friend's model? Be as specific as possible.

$\beta_0 = 2$ and $\beta_1 = -1$.

Since the two models are equivalent, this implies that

$$\beta_0 = \hat{\alpha}_{orange0} - \hat{\alpha}_{apple0}$$

and

$$\beta_1 = \hat{\alpha}_{orange1} - \hat{\alpha}_{apple1}$$

Therefore, the coefficient estimates in the softmax model are also 2 and -1 respectively for x^0 and x^1 respectively.

(d)

Now suppose that you and your friend fit the same two models on a different data set. This time, your friend gets the coefficient estimates $\hat{\alpha}_{orange0}$ = 1.2, $\hat{\alpha}_{orange1}$ = -2, $\hat{\alpha}_{apple0}$ = 3, $\hat{\alpha}_{apple1}$ = 0.6. What are the coefficient estimates in your model?

The coefficient estimates in my model are:

β_0 = 1.2 - 3 = -1.8

and

β_1 = -2 - 0.6 = -2.6

(e)

Finally, suppose you apply both models from (d) to a data set with 2,000 test observations. What fraction of the time do you expect the predicted class labels from your model to agree with those from your friend's model? Explain your answer.

Always since the models are equivalent and the classification predictions will not differ.

Bonus Exercise

Theorem: as n $\rightarrow \infty$, 1-NN prediction error is no more than twice of the error of the Bayes optimal classifier.

Bayes Optimal Predictor

- Assume our data is collected in an i.i.d fashion, i.e., $(x, y) \sim$ P (say y $\in [-1, 1]$)
- Assume we know P(y | x) for now

- Q: What label you would predict?
- A: We will simply predict the most-likely label,
- Bayes Optimal Predictor is $h_{opt}(x) = argmax P(y|x)\ y \in [-1, 1]$

Example:

$$\left\{ \begin{matrix} P(1|x) = 0.8 \\ P(-1|x) = 0.2 \end{matrix} \right\}$$

$y_b := h_{opt}(x) = 1$

Q: What's the probability of h_{opt} making a mistake on x?

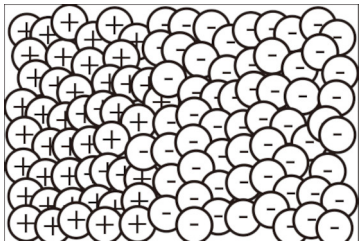
A: $\epsilon_{opt} = 1 - P(y_b|x) = 0.2$

Assume $x \in [-1, 1]^2$, P(x) has support everywhere P(x) > 0, $\forall x \in [-1, 1]^2$

P(x,y)

P(x) = P(x, +1) + P(x,-1)

What does it look like when $n \to \infty$?



Given test x, as $n \to \infty$, its nearest neighbor xNN is super close, i.e., $d(x, xNN) \to 0$!

Proof: 1. Fix a test example x, denote its NN as xNN. When $n \to \infty$, we have $xNN \to x$

2. Without loss of generality (WLOG) assume for x, the Bayes optimal predicts $y_b = h_{opt}(x)$ = 1
3. Calculate the 1-NN's prediction error:
    - Case 1 when yNN = 1 (it happens w/ prob P(1 | xNN) = P(1 | x)):
        - The probability of making a mistake: $\epsilon = P(y \neq 1|x) = P(y = -1|x) = 1 - P(y_b|x)$
    - Case 2 when yNN = -1 (it happens w/ prob P(-1 | xNN) = P(-1 | x)):
        - The probability of making a mistake: $\epsilon = P(y \neq -1|x) = P(y = 1|x) = P(y_b|x)$

4. Our prediction error at x:

$$
\begin{aligned}
&P(1|x)(1-P(y_b|x)+P(-1|x)P(y_b|x)\\
&P(1|x) <= 1\\
&P(y_b|x) <= 1\\
&P(-1|x) = 1-P(y_b|x) (\because y_b = 1)\\
&\implies P(1|x)(1-P(y_b|x)+P(-1|x)P(y_b|x) <= (1-P(y_b|x))+(1-P(y_b|x))\\
&\because 1-P(y_b|x) = \epsilon_{opt}\\
&\implies P(1|x)(1-P(y_b|x)+P(-1|x)P(y_b|x) <= 2\epsilon_{opt}
\end{aligned}
$$

Reference: 1. https://www.cs.cornell.edu/courses/cs4780/2022fa/slides/KNN_annotated.pdf

```
allDone();
```

```
<IPython.lib.display.Audio object>
```