

# DATA 221 Homework 2

Chris Low

## Problem 1

### 1(a)

They are **ordinal categorical variables**. They are categories with a meaningful order (e.g., Worsened less than Unchanged less than Improved). There is a natural order but not we do not have a numerical sense to it.

### 1(b)

```
import pandas as pd
import statsmodels.api as sm

train_df = pd.read_csv('../hw2/data/PCC_study_train.csv')

# (One-Hot Encoding)
X_cat = pd.get_dummies(train_df[['WBscore', 'PCCsymp']], drop_first=True)
X_cat = X_cat.astype(float)
X_cat = sm.add_constant(X_cat)

y = train_df['vax_status'].map({'Unvaccinated': 0, 'Vaccinated': 1})

# Train Model
log_reg_cat = sm.Logit(y, X_cat).fit()

# View Summary to find significant variables
print(log_reg_cat.summary())
```

Optimization terminated successfully.

Current function value: 0.408057

Iterations 6

#### Logit Regression Results

Dep. Variable:	vax_status	No. Observations:	75			
Model:	Logit	Df Residuals:	70			
Method:	MLE	Df Model:	4			
Date:	Wed, 28 Jan 2026	Pseudo R-squ.:	0.4106			
Time:	18:33:07	Log-Likelihood:	-			
30.604						
converged:	True	LL-Null:	-			
51.926						
Covariance Type:	nonrobust	LLR p-value:	1.227e-			
08						
=====						
	coef	std err	z	P> z	[0.025	0.975]
-----						
const	1.9769	0.521	3.791	0.000	0.955	2.999
WBscore_Unchanged	-2.0144	0.755	-2.668	0.008	-3.494	-
0.535						
WBscore_Worsened	-1.0089	1.012	-0.997	0.319	-2.993	0.975
PCCsymp_More	-2.4890	1.069	-2.327	0.020	-4.585	-
0.393						
PCCsymp_Same	-2.7364	0.836	-3.275	0.001	-4.374	-
1.099						
=====						

If we treat WBscore and PCCsymp as categorical variables, we find that **WBscore\_Unchanged**, **PCCsymp\_More**, and **PCCsymp\_Same** are statistically significant predictors of vaccination status, while **WBscore\_Worsened** is not significant.

### 1(c)

Instead of creating many dummy columns, like we did in (b), we are creating two numeric columns, WBscore\_num and PCCsymp\_num.

```
train_df = pd.read_csv('../hw2/data/PCC_study_train.csv')

y = train_df['vax_status'].map({'Unvaccinated': 0, 'Vaccinated': 1})
```

```

wb_map = {'Worsened': 0, 'Unchanged': 1, 'Improved': 2}
pcc_map = {'More': 0, 'Same': 1, 'Less': 2}

train_df['WBscore_num'] = train_df['WBscore'].map(wb_map)
train_df['PCCsymp_num'] = train_df['PCCsymp'].map(pcc_map)

X_num = train_df[['WBscore_num', 'PCCsymp_num']]
X_num = sm.add_constant(X_num)

log_reg_num = sm.Logit(y, X_num).fit()

print(log_reg_num.summary())

```

Optimization terminated successfully.

Current function value: 0.466680

Iterations 6

Logit Regression Results						
=====						
Dep. Variable:	vax_status	No. Observations:	75			
Model:	Logit	Df Residuals:	72			
Method:	MLE	Df Model:	2			
Date:	Wed, 28 Jan 2026	Pseudo R-squ.:	0.3259			
Time:	18:33:07	Log-Likelihood:	-			
35.001						
converged:	True	LL-Null:	-			
51.926						
Covariance Type:	nonrobust	LLR p-value:	4.462e-			
08						
=====						
	coef	std err	z	P> z	[0.025	0.975]
-----						
const	-3.6377	0.890	-4.085	0.000	-5.383	-
1.892						
WBscore_num	0.9682	0.497	1.948	0.051	-0.006	1.942
PCCsymp_num	1.5780	0.552	2.859	0.004	0.496	2.660
=====						

If we treat WBscore and PCCsymp as numeric variables, we find that PCCsymp is a statistically significant predictor of vaccination status, while WBscore is not significant at the 5% level. (Alternatively, we can say that WBscore is marginally significant ( $p \approx 0.05$ ), while PCCsymp is clearly significant.)

### 1(d)

```
from sklearn.metrics import roc_curve, auc

test_df = pd.read_csv('../hw2/data/PCC_study_test.csv')

y_test = test_df['vax_status2'].map({'Unvaccinated': 0, 'Vaccinated': 1})

# Model from 1(b):
X_test_cat = pd.get_dummies(
    test_df[['WBscore', 'PCCsymp']],
    drop_first=True
)

X_test_cat = X_test_cat.reindex(
    columns=X_cat.columns.drop('const'),
    fill_value=0
)

X_test_cat = sm.add_constant(X_test_cat)
X_test_cat = X_test_cat.astype(float)

# Using train log_reg_cat to predict
y_prob_cat = log_reg_cat.predict(X_test_cat)

fpr_cat, tpr_cat, _ = roc_curve(y_test, y_prob_cat)
auc_cat = auc(fpr_cat, tpr_cat)

print("Categorical AUC:", auc_cat)
```

Categorical AUC: 0.8828125

```
wb_map = {'Worsened': 0, 'Unchanged': 1, 'Improved': 2}
pcc_map = {'More': 0, 'Same': 1, 'Less': 2}

test_df['WBscore_num'] = test_df['WBscore'].map(wb_map)
test_df['PCCsymp_num'] = test_df['PCCsymp'].map(pcc_map)

X_test_num = test_df[['WBscore_num', 'PCCsymp_num']]
X_test_num = sm.add_constant(X_test_num)
```

```

y_prob_num = log_reg_num.predict(X_test_num)

fpr_num, tpr_num, _ = roc_curve(y_test, y_prob_num)
auc_num = auc(fpr_num, tpr_num)

print("Numerical AUC:", auc_num)

```

Numerical AUC: 0.8627232142857143

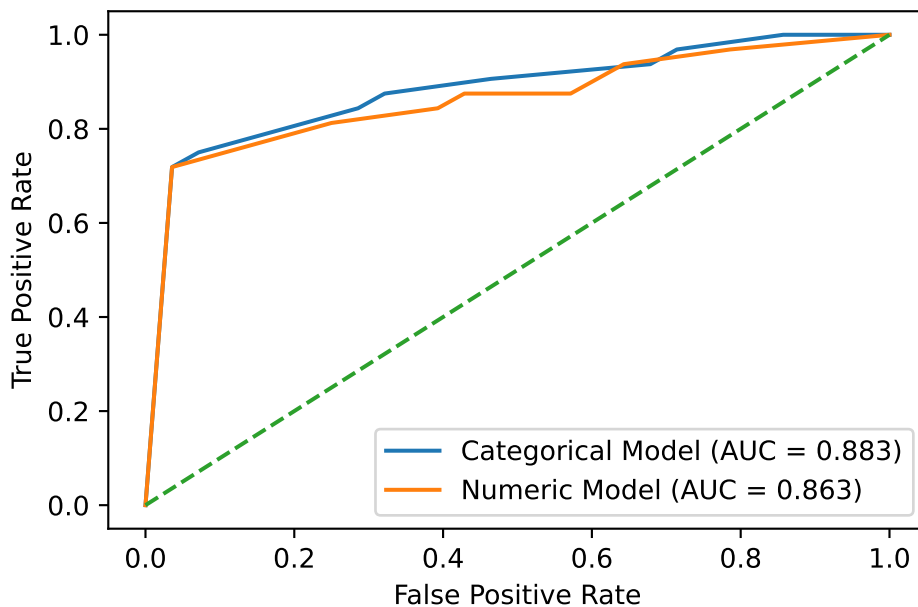
```

import matplotlib.pyplot as plt

plt.figure()
plt.plot(fpr_cat, tpr_cat, label=f'Categorical Model (AUC = {auc_cat:.3f})')
plt.plot(fpr_num, tpr_num, label=f'Numeric Model (AUC = {auc_num:.3f})')
plt.plot([0,1], [0,1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()

print("AUC (categorical):", auc_cat)
print("AUC (numeric):", auc_num)

```



```
AUC (categorical): 0.8828125
AUC (numeric): 0.8627232142857143
```

I recommend the **Categorical Model**. It achieves a higher AUC, which means that it captures non-linear relationships between the specific categories (like “Unchanged” vs “Improved”) better than the linear assumption of the numeric model.

## Problem 2

### 2(a)

```
# Prep data: pasted from above
train_df = pd.read_csv('../hw2/data/PCC_study_train.csv')

y_train = train_df['vax_status'].map({'Unvaccinated': 0, 'Vaccinated': 1})

wb_map = {'Worsened': 0, 'Unchanged': 1, 'Improved': 2}
pcc_map = {'More': 0, 'Same': 1, 'Less': 2}

train_df['WBscore_num'] = train_df['WBscore'].map(wb_map)
train_df['PCCsymp_num'] = train_df['PCCsymp'].map(pcc_map)

X_train = train_df[['WBscore_num', 'PCCsymp_num']]
```

I will play around with gini and entropy, keeping max\_depth at 3 to prevent overfitting.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree

tree = DecisionTreeClassifier(
    criterion='gini',
    max_depth=3,
    random_state=42
)

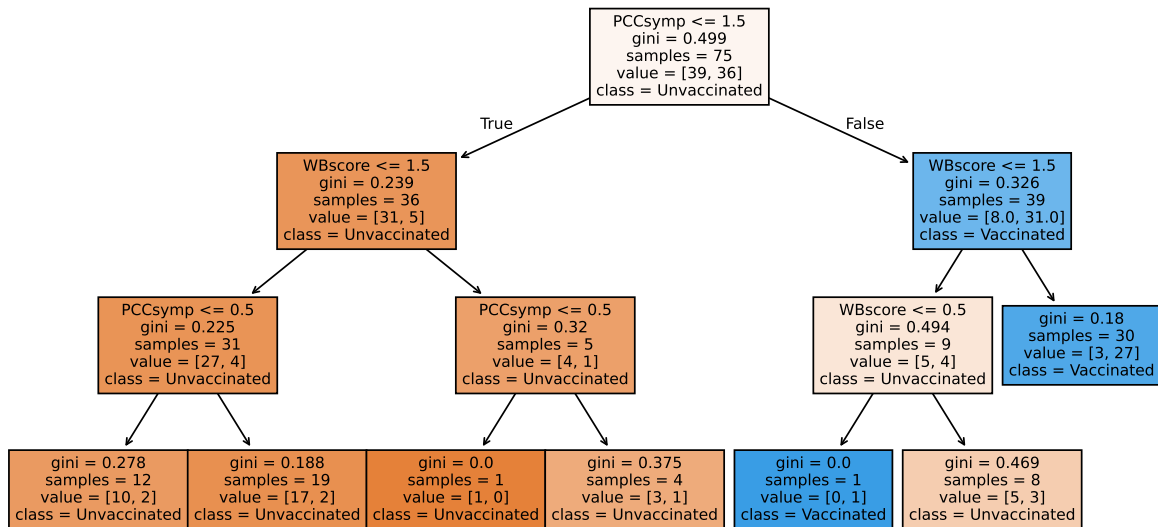
tree.fit(X_train, y_train)

plt.figure(figsize=(12,6))
plot_tree(
```

```

tree,
feature_names=['WBscore', 'PCCsymp'],
class_names=['Unvaccinated', 'Vaccinated'],
filled=True
)
plt.show()

```



```

test_df = pd.read_csv('../hw2/data/PCC_study_test.csv')

y_test = test_df['vax_status2'].map({'Unvaccinated': 0, 'Vaccinated': 1})

test_df['WBscore_num'] = test_df['WBscore'].map(wb_map)
test_df['PCCsymp_num'] = test_df['PCCsymp'].map(pcc_map)

X_test = test_df[['WBscore_num', 'PCCsymp_num']]

y_prob_tree = tree.predict_proba(X_test)[:, 1]

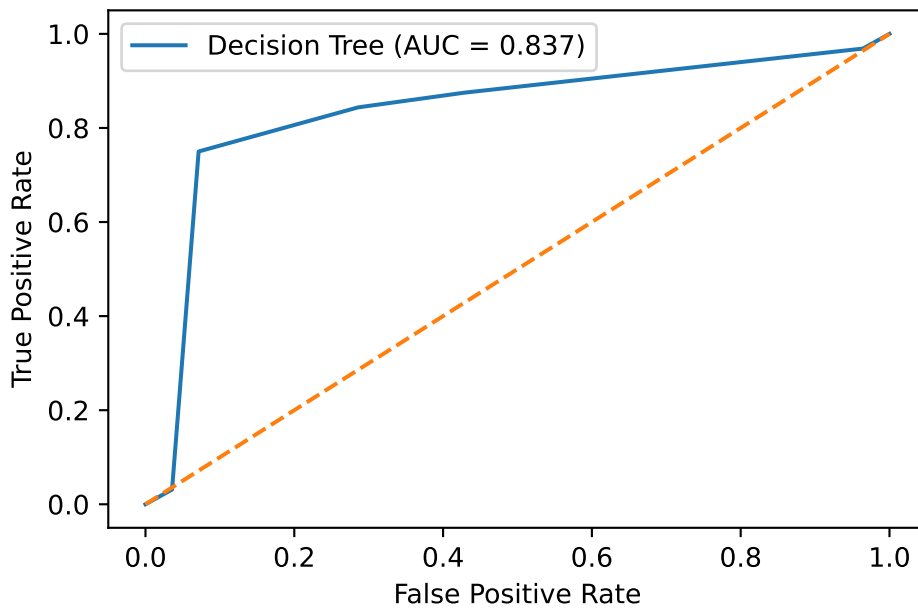
fpr_tree, tpr_tree, _ = roc_curve(y_test, y_prob_tree)
auc_tree = auc(fpr_tree, tpr_tree)

plt.figure()
plt.plot(fpr_tree, tpr_tree, label=f'Decision Tree (AUC = {auc_tree:.3f})')
plt.plot([0,1], [0,1], linestyle='--')

```

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()

print("Decision Tree AUC:", auc_tree)
```



Decision Tree AUC: 0.8370535714285715

Now I will try entropy with `max_depth = 5`:

```
dt_model = DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=42)
dt_model.fit(X_train, y_train)

plt.figure(figsize=(12,6))
plot_tree(
    dt_model,
    feature_names=['WBscore', 'PCCsymp'],
    class_names=['Unvaccinated', 'Vaccinated'],
    filled=True
)
plt.show()
```

```

test_df = pd.read_csv('../hw2/data/PCC_study_test.csv')

y_test = test_df['vax_status2'].map({'Unvaccinated': 0, 'Vaccinated': 1})

test_df['WBscore_num'] = test_df['WBscore'].map(wb_map)
test_df['PCCsymp_num'] = test_df['PCCsymp'].map(pcc_map)

X_test = test_df[['WBscore_num', 'PCCsymp_num']]

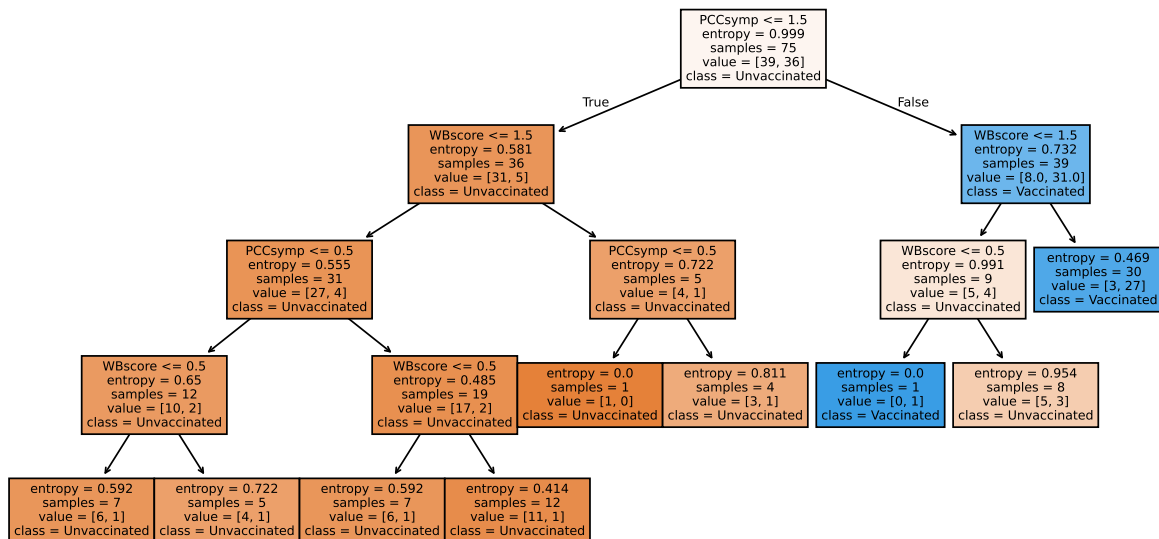
y_prob_tree = dt_model.predict_proba(X_test)[: , 1]

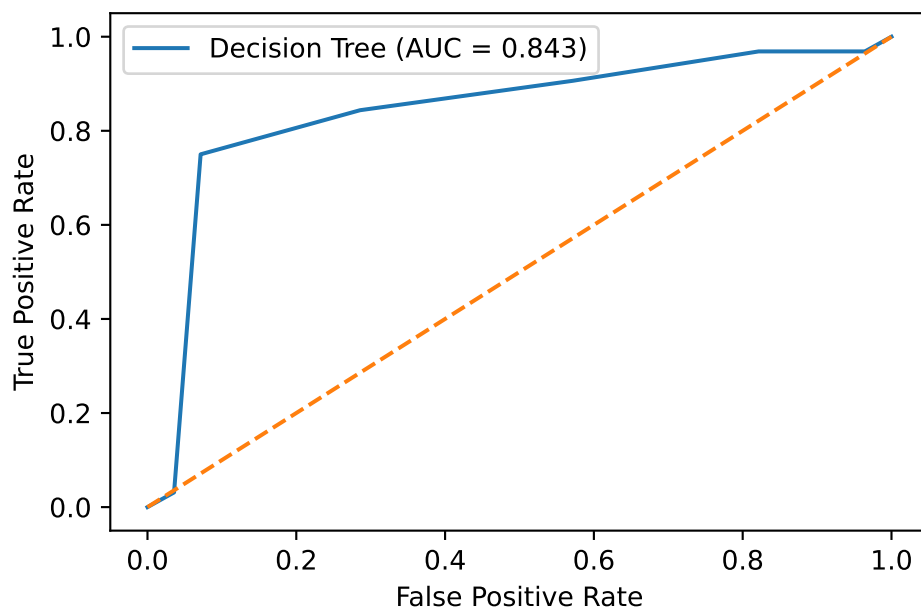
fpr_tree, tpr_tree, _ = roc_curve(y_test, y_prob_tree)
auc_tree = auc(fpr_tree, tpr_tree)

plt.figure()
plt.plot(fpr_tree, tpr_tree, label=f'Decision Tree (AUC = {auc_tree:.3f})')
plt.plot([0,1], [0,1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()

print("Decision Tree AUC (Entropy):", auc_tree)

```





Decision Tree AUC (Entropy): 0.8431919642857144