EC3357:Machine Learning

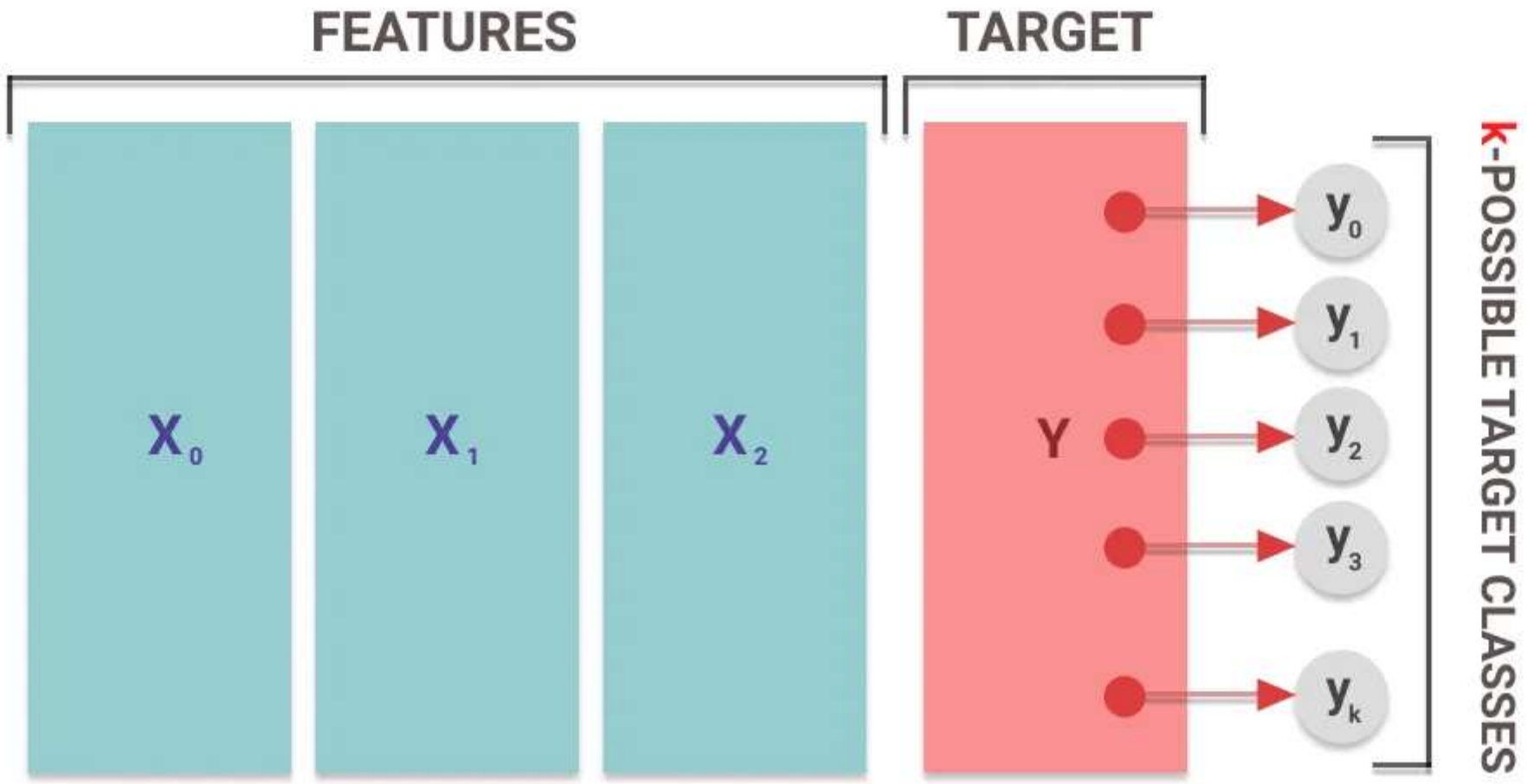
Lecture 4B: Multinomial Logistic Regression

What is Multinomial Logistic Regression?

- Multinomial Logistic Regression is the regression analysis to conduct when the dependent variable is nominal with more than two levels.
- It is an extension of logistic regression that adds native support for multi-class classification problems.
- Binomial Logistic Regression:
  - Standard logistic regression that predicts a binomial probability (i.e. for two classes) for each input example.
- Multinomial Logistic Regression:
  - Modified version of logistic regression that predicts a multinomial probability (i.e. more than

Example:

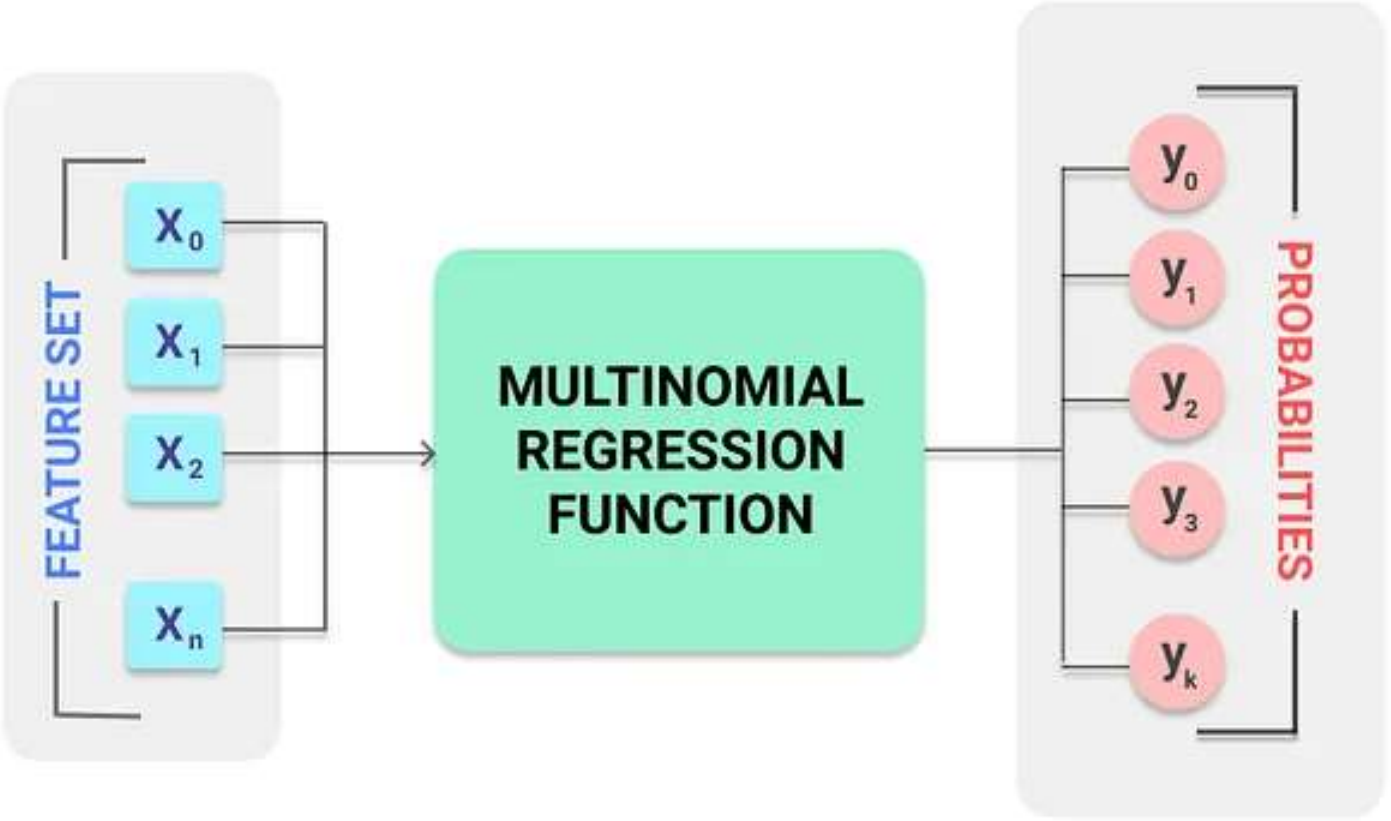
- Assume that we observed a class of students for a whole term. At the end of the term, we gave each student a computer game as a gift for their effort. Each participant was free to choose between three games – an action, a puzzle or a sports game. We want to know how students' scores in math, reading, and writing affect their choice of game. Note that the choice of the game is a nominal dependent variable with three levels. Therefore, multinomial regression is an appropriate analytic approach to this scenario.



The outcome predicted for a feature set is one of the k-possible outcomes

How does MLR work?

- The multinomial regression function is a statistical classification algorithm.
- What this means is that once we feed the function a set of features, the model performs a series of mathematical operations to normalize the input values into a vector of values that follows a probability distribution.



The MLR function calculates probabilities for possible target classes from the given feature set

Step 1: Gather your data

- To start with a simple example, let’s say that your goal is to build a logistic regression model in Python in order to determine the category of employees based on salary and years of experience.
- Here, there are 3 possible outcomes: Executive (represented by the value of ‘2’), managerial (represented by the value of ‘1’) and operational (represented by the value of ‘0’)

Step 2: Import the needed Python packages

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,
confusion_matrix, classification_report
```

Step 3: Build a dataframe

```
data = {
    'salary': [50000, 60000, 80000, 120000, 30000, 70000, 40000, 100000, 150000, 20000],
    'years_of_experience': [1, 2, 5, 10, 0.5, 3, 1.5, 7, 12, 0.2],
    'category': [0, 0, 1, 2, 0, 1, 0, 1, 2, 0]
}
df = pd.DataFrame(data)
print (df)
```

Category is the target variable, ranging from 0 to 2

	salary	years_of_experience	category
0	50000	1.0	0
1	60000	2.0	0
2	80000	5.0	1
3	120000	10.0	2
4	30000	0.5	0
5	70000	3.0	1
6	40000	1.5	0
7	100000	7.0	1
8	150000	12.0	2
9	20000	0.2	0

Step 4: Create the logistic regression in Python

- Set the independent variables (represented as X) and the dependent variable (represented as y):

```
X = df[['salary', 'years_of_experience']]
y = df['category']
```

- Split the dataset into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

- Standardize features by removing the mean and scaling to unit variance

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

- Create and train the logistic regression model

```
    model =
LogisticRegression(multi_class='multinomial’,
    solver='lbfgs', max_iter=1000)
    model.fit(X_train, y_train)
```

Step 5: Make predictions on the test set and evaluate the model

- Make predictions on the test set

```
y_pred = model.predict(X_test)
```

- Evaluate the model

```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
```

```
Example prediction
    new_data = np.array([[75000, 10]])  # New
staff with 75,000    salary and 10 years of
experience
    new_data = scaler.transform(new_data)
    predicted_category = model.predict(new_data)
    print(f"Predicted Category for new data:
    {predicted_category[0]}")
```