WEEK 5

Classification: Logistic Regression & Beyond

Dichotomous and Polytomous Dependent Variables

ECO447 — Machine Learning for Economists

Assoc. Prof. Dr. Onur Polat
Hacettepe University, Institute of Informatics

Today's Outline

1. Binary Classification in Economics
2. Logistic Regression: Theory and Estimation
3. Maximum Likelihood Estimation
4. Model Evaluation for Classifiers
5. Multinomial and Ordinal Logistic Regression
6. Probit and Other Link Functions
7. Application: Credit Default Prediction

01

Binary Classification in Economics

Economic Classification Problems

- Credit default: will a borrower default on their loan? (Yes/No)
- Recession prediction: is the economy entering a recession? (Yes/No)
- Bank failure: will a bank fail in the next year? (Yes/No)
- Trade direction: will exports increase or decrease? (Up/Down)
- Policy adoption: will a country adopt a carbon tax? (Yes/No)
- Employment: is an individual employed or unemployed?
- Market regime: bull market or bear market?

Why Not Linear Regression for Classification?

- Linear Probability Model (LPM): $Y = X\beta + \varepsilon$ where $Y \in \{0,1\}$
- Problem 1: predicted probabilities can be < 0 or > 1
- Problem 2: heteroscedastic errors — $\text{Var}(\varepsilon|X) = P(1-P)$
- Problem 3: marginal effects are constant (unrealistic)
- LPM advantage: easy to interpret, quick estimation
- LPM is still used in economics (Angrist & Pischke argue for it in causal settings)
- But for prediction, logistic regression is generally preferred

02

Logistic Regression

Logistic Regression Model

$$P(Y=1|X) = \sigma(X\beta) = 1 / (1 + e^{(-X\beta)})$$

Sigmoid function ensures probabilities stay in [0,1]

$$\log(P/(1-P)) = X\beta \quad \text{(log-odds / logit)}$$

Log-odds is linear in X — the logit link function

$$\sigma(z) \text{ maps } \mathbb{R} \to (0,1): \text{ sigmoid function}$$

S-shaped curve: flat at extremes, steep in the middle

$$\text{Marginal effect: } \partial P/\partial x_j = \beta_j \cdot P(1-P)$$

Interpreting Logistic Regression

- Coefficients β_j: change in log-odds for unit increase in x_j
- Odds ratio $\exp(\beta_j)$: multiplicative change in odds
- Example: $\beta = 0.5 \rightarrow$ odds multiply by $e^{0.5} \approx 1.65$ (65% increase in odds)
- Marginal effects at means (MEM): evaluate $\partial P/\partial x$ at mean values
- Average marginal effects (AME): average $\partial P/\partial x$ over all observations
- AME is generally preferred — more robust to functional form
- In ML: we often care about prediction, not coefficient interpretation

03

Maximum Likelihood Estimation

MLE for Logistic Regression

$$L(\beta) = \prod_i P(x_i)^{y_i} \cdot (1-P(x_i))^{(1-y_i)}$$

Likelihood function: probability of observing the data

$$\ell(\beta) = \sum_i [y_i \log P(x_i) + (1-y_i) \log(1-P(x_i))]$$

Log-likelihood: easier to optimize (sum instead of product)

$$\hat{\beta}_{MLE} = \text{argmax}_\beta \ \ell(\beta)$$

Maximize log-likelihood = minimize negative log-likelihood (cross-entropy loss)

$$\partial\ell/\partial\beta = X'(y - \hat{P}) = 0 \ \rightarrow \ \text{no closed form}$$

PYTHON

Logistic Regression in Python

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (classification_report,
    confusion_matrix, roc_auc_score, roc_curve)
# Fit logistic regression
logreg = LogisticRegression(max_iter=1000, random_state=42)
logreg.fit(X_train, y_train)
# Predict probabilities and classes
y_prob = logreg.predict_proba(X_test)[:, 1]
y_pred = logreg.predict(X_test)
# Evaluation
print(classification_report(y_test, y_pred))
print(f"AUC-ROC: {roc_auc_score(y_test, y_prob):.4f}")
# Coefficients as odds ratios
import pandas as pd
odds = pd.DataFrame({
    "Feature": feature_names,
    "Coef": logreg.coef_[0],
    "Odds Ratio": np.exp(logreg.coef_[0])
}).sort_values("Odds Ratio", ascending=False)
```

- predict_proba gives probabilities, predict gives class labels
- AUC-ROC is threshold-independent evaluation metric

04

Classification Evaluation

The Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Key Classification Metrics

- Accuracy = (TP+TN) / (TP+TN+FP+FN) — misleading with imbalanced data
- Precision = TP / (TP+FP) — of predicted positives, how many correct?
- Recall = TP / (TP+FN) — of actual positives, how many detected?
- F1 = 2 · (Precision · Recall) / (Precision + Recall) — harmonic mean
- AUC-ROC: area under TPR vs FPR curve — threshold-independent
- Economic context: cost of FP vs FN is usually asymmetric
- Credit scoring: missing a default (FN) is much costlier than false alarm (FP)

PYTHON

ROC Curve and Threshold Selection

```
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# Compute ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
auc = roc_auc_score(y_test, y_prob)

# Plot
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"Logistic Reg (AUC={auc:.3f})")
plt.plot([0,1], [0,1], "k--", label="Random")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()

# Optimal threshold (Youden's J statistic)
j_scores = tpr - fpr
optimal_idx = j_scores.argmax()
optimal_threshold = thresholds[optimal_idx]
print(f"Optimal threshold: {optimal_threshold:.3f}")
```

- AUC = 0.5: random, AUC = 1.0: perfect classifier
- Threshold choice depends on relative cost of FP vs FN

05

Multinomial & Ordinal Models

Multinomial Logistic Regression

- Y has K > 2 unordered categories (e.g., sector choice, vote choice)
- $P(Y=k|X) = \exp(X\beta_k) / \Sigma_j \exp(X\beta_j)$ — softmax function
- One category is the reference (baseline): K-1 sets of coefficients
- Log-odds relative to reference: $\log(P(Y=k)/P(Y=ref)) = X\beta_k$
- IIA assumption: independence of irrelevant alternatives (limitation)
- Alternative: nested logit, mixed logit (relaxes IIA)
- Scikit-learn: LogisticRegression(multi_class='multinomial')

Ordinal Logistic Regression

- Y has ordered categories (e.g., credit rating: D < C < B < A < AA < AAA)
- Proportional odds model: $\log(P(Y \leq k)/P(Y > k)) = \alpha_k - X\beta$
- Single set of β coefficients — same effect across cutpoints
- Cutpoints α_k define the thresholds between categories
- Proportional odds assumption: parallel regression assumption
- Test: Brant test for proportional odds
- Python: statsmodels OrderedModel, mord library

06

Application: Credit Scoring

PYTHON

Credit Default Prediction Pipeline

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer, f1_score

# Features: financial ratios, demographics
features = ["debt_income", "credit_util", "num_late",
            "employment_years", "loan_amount", "income"]
# Pipeline with scaling
pipe = Pipeline([
    ("scaler", StandardScaler()),
    ("classifier", LogisticRegression(
        class_weight="balanced",  # handle imbalance
        C=1.0,                    # regularization
        max_iter=1000))
])
# Evaluate with F1 (handles class imbalance)
scores = cross_val_score(pipe, X_train, y_train,
    cv=5, scoring="f1")
print(f"CV F1: {scores.mean():.4f} ± {scores.std():.4f}")
```

- class_weight='balanced' adjusts for imbalanced default rates
- C parameter controls regularization strength (lower = more)

07

Probit & Alternative Link Functions

The Probit Model

- $P(Y=1|X) = \Phi(X\beta)$ where Φ is the standard normal CDF
- Latent variable interpretation: $Y^* = X\beta + \varepsilon$, $Y = 1$ if $Y^* > 0$
- Error ε follows standard normal distribution (vs. logistic for logit)
- Favored in classical econometrics; dominant in discrete choice models (McFadden, 1974)
- Coefficients not directly interpretable → report marginal effects at means or AMEs
- Standard reference in labor economics (Heckman selection), health, and voting studies

Logit vs Probit: When Does It Matter?

- Logistic distribution has fatter tails → more weight on extreme observations
- Rule of thumb: logit coefficients ≈ probit coefficients × 1.6 ($\pi/\sqrt{3}$)
- For predicted probabilities in [0.1, 0.9], the two models agree to within ~1 pp
- Choice is largely conventional: logit in CS/ML, probit in econometrics and panel data
- Probit is required when assuming jointly normal errors (e.g., bivariate probit, Heckman)
- For prediction-focused ML pipelines, logistic regression is the default

Complementary Log-Log and Beyond

- cloglog link: $\log(-\log(1-P)) = X\beta$ — asymmetric, used for rare events
- Natural fit for discrete-time hazard models (survival analysis, time-to-default)
- Example: King-Zeng (2001) rare-event logit corrects downward bias when P(Y=1) is small
- Cauchit link: heavy-tailed, robust to outliers in the latent variable
- Firth correction (penalized MLE): small-sample bias correction, separable data
- In Python: statsmodels GLM with family=Binomial() + link=cloglog() / probit() / cauchit()

08

Regularized Logistic Regression

Why Regularize?

- High-dimensional X: text features, macro factor zoo, firm fundamentals — p can exceed n
- Perfect separation: MLE coefficients → $\pm\infty$ when classes are linearly separable
- Multicollinearity among macroeconomic predictors inflates coefficient variance
- Regularization: add penalty $\lambda \cdot R(\beta)$ to the negative log-likelihood
- L2 (Ridge): $R(\beta) = \Sigma\beta^2_j$ — shrinks all coefficients, keeps all variables
- L1 (LASSO): $R(\beta) = \Sigma|\beta_j|$ — performs variable selection, sets some coefficients to 0
- Elastic Net: $\alpha L1 + (1-\alpha)L2$ — compromise for correlated predictors

PYTHON

Regularized Logistic in scikit-learn

```
from sklearn.linear_model import LogisticRegressionCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

# Elastic-net logistic with CV-selected C and l1_ratio
pipe = Pipeline([
    ("scaler", StandardScaler()),
    ("logit",  LogisticRegressionCV(
        penalty="elasticnet", solver="saga",
        l1_ratios=[0.1, 0.5, 0.9],
        Cs=[0.01, 0.1, 1.0, 10.0],
        cv=5, scoring="roc_auc",
        max_iter=5000, random_state=42))
])
pipe.fit(X_train, y_train)
print("Best C:",        pipe["logit"].C_[0])
print("Best l1_ratio:", pipe["logit"].l1_ratio_[0])
print("Non-zero vars:", (pipe["logit"].coef_[0] != 0).sum())
```

- ALWAYS standardize X before regularization — the penalty is scale-sensitive
- saga solver supports L1, L2 and elastic-net; liblinear for L1 only on small data

09

Imbalanced Classification

The Class Imbalance Problem

- Economic rare events: bank failures (~1%), mortgage defaults (~2-5%), recessions (~10%)
- A model predicting "no default" for everyone scores 98% accuracy — and is useless
- Default decision threshold of 0.5 is arbitrary when base rate is 2%
- Three families of fixes: (1) resampling, (2) cost-sensitive learning, (3) threshold tuning
- Resampling: SMOTE oversamples minority with synthetic examples; random under-samples majority
- Cost-sensitive: class_weight="balanced" scales the loss inversely to class frequency
- Report precision, recall, F1, precision-recall AUC — not just accuracy and ROC-AUC

PYTHON

Handling Imbalance: SMOTE and Class Weights

```
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline as ImbPipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import average_precision_score

# SMOTE applied ONLY to training folds (inside the pipeline)
pipe = ImbPipeline([
    ("smote", SMOTE(random_state=42, k_neighbors=5)),
    ("logit", LogisticRegression(max_iter=2000))
])
pipe.fit(X_train, y_train)
y_prob = pipe.predict_proba(X_test)[:, 1]

# PR-AUC is the right metric under heavy imbalance
print(f"PR-AUC: {average_precision_score(y_test, y_prob):.4f}")

# Alternative: cost-sensitive weighting
cs = LogisticRegression(class_weight="balanced", max_iter=2000)
```

- Never resample the test set — it must mirror the real-world class distribution
- ROC-AUC can be optimistic under extreme imbalance; prefer PR-AUC and lift curves

10

Real-World Economic Applications

Application 1: Mortgage Default & Credit Scoring

- FICO score, DTI ratio, LTV, loan age, employment years → predict P(default) within 12 months
- Fannie Mae and Freddie Mac loan performance data: millions of observations, panel structure
- Logistic regression remains the production workhorse under Basel/CECL regulatory review
- Model must be interpretable: regulators demand per-variable justification of pricing decisions
- Fair lending constraints (ECOA, HMDA): disparate impact tested across protected classes
- Modern practice: logistic as a benchmark; gradient boosting for scoring with SHAP explanations

Application 2: Early Warning Systems for Banking Crises

- Reinhart-Rogoff style EWS: predict systemic banking crisis within h quarters ahead
- Features: credit-to-GDP gap, house price growth, current account deficit, foreign debt share
- Logistic regression + rolling window estimation = BCBS countercyclical capital buffer tool
- Danielsson, de Haan et al.: cross-country panel logit on 60+ economies, 1970–present
- Trade-off: tighter threshold catches more crises (recall) but more false alarms (precision)
- Policy cost asymmetry: a missed crisis dwarfs the output cost of a false alarm

Application 3: Fraud and Money-Laundering Detection

- Transaction-level data: amount, time, merchant category, geo-location, device, velocity features
- Base rate of fraud < 0.1% → extreme imbalance, cost-sensitive logistic is a strong baseline
- Online learning variants (SGD logistic) scale to billions of transactions per day
- False positives have real cost: declined legitimate transactions damage customer trust
- Regulatory angle: AML/CFT reports (FINCEN SAR filings) require audit-trail explanations
- Combined with graph features (entity resolution, shared counterparties) in modern stacks

Application 4: Discrete Choice in Household Economics

- Mode choice: car vs. bus vs. metro — multinomial logit (McFadden Nobel 2000)
- Labor force participation: binary logit on female LFP with children, education, husband income
- Occupational choice: multinomial logit over blue-collar / white-collar / self-employed
- Vote choice: multinomial logit over parties, with voter demographics and policy positions
- Random utility foundation: $U_{ij} = V_{ij} + \varepsilon_{ij}$, $\varepsilon_{ij} \sim$ i.i.d. Gumbel → logit choice probabilities
- Nested logit, mixed logit (random coefficients) relax IIA when alternatives share features

11

Case Study — HMDA Mortgage Approval

HMDA: The Home Mortgage Disclosure Act Dataset

- HMDA (CFPB): every US mortgage application since 1990 — over 20 million rows per year
- Target Y = 1 if loan approved, 0 if denied (or withdrawn / incomplete, with caveats)
- Features: loan amount, applicant income, LTV, DTI, property type, race, gender, census tract
- A canonical fair-lending dataset: used in Munnell et al. (1996) Boston Fed study on discrimination
- Modern academic use: Bartlett, Morse, Stanton, Wallace (2022) in J. Fin. Econ. — algorithmic bias
- Public download: https://ffiec.cfpb.gov/data-browser/data/

PYTHON

HMDA Case Study — Modeling Pipeline

```
import pandas as pd, numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

df = pd.read_csv("hmda_2022_sample.csv")
y = (df["action_taken"] == 1).astype(int)   # 1 = originated

num = ["loan_amount", "income", "loan_to_value_ratio",
       "debt_to_income_ratio", "property_value"]
cat = ["loan_type", "property_type", "occupancy"]
X = pd.get_dummies(df[num + cat], columns=cat, drop_first=True, dtype=int)

Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.3,
                                      stratify=y, random_state=42)
pipe = Pipeline([("sc", StandardScaler()),
                 ("lr", LogisticRegression(max_iter=1000,
                        class_weight="balanced"))]).fit(Xtr, ytr)
```

- drop_first=True avoids the dummy-variable trap; dtype=int keeps columns numeric
- stratify=y preserves the approval rate across train/test splits

PYTHON

HMDA Case Study — Evaluation & Fair-Lending Audit

```
from sklearn.metrics import (roc_auc_score, classification_report,
                             confusion_matrix)
y_prob = pipe.predict_proba(Xte)[:, 1]
y_pred = (y_prob >= 0.5).astype(int)

print(f"AUC: {roc_auc_score(yte, y_prob):.4f}")
print(classification_report(yte, y_pred))

# Approval rate audit by applicant race (disparate impact)
audit = (df.loc[yte.index]
          .assign(pred=y_pred)
          .groupby("derived_race")["pred"].mean()
          .sort_values(ascending=False))
print(audit)

# 80% rule: minority approval / majority approval must be >= 0.8
ratio = audit.min() / audit.max()
print(f"Disparate-impact ratio: {ratio:.3f}  (>= 0.80 safe harbor)")
```

- Fair-lending compliance tests go beyond accuracy — a well-calibrated model can still fail
- Removing race from X does NOT remove disparate impact — proxies (zip code) remain

Case Study Takeaways & Regulatory Lessons

- Logistic regression gives both P(approve) and the per-variable contributions regulators demand
- A model with AUC = 0.85 can still violate the 80% rule — predictive accuracy ≠ fairness
- Protected variables excluded from X are inferred by proxies (zip code, income, surname)
- Document the model: training data vintage, feature set, cutoff, periodic re-estimation
- CFPB & OCC guidance (SR 11-7): model risk management — development, implementation, validation
- Open research: counterfactual fairness, equalized odds post-processing, two-sided calibration

SUMMARY

Week 5 Key Takeaways

- Logistic regression maps linear predictors to probabilities via sigmoid
- Estimated by MLE (no closed form) — equivalent to minimizing cross-entropy
- Coefficients interpreted as log-odds ratios, or via marginal effects
- Classification evaluation: use AUC-ROC, F1, precision/recall — not just accuracy
- Multinomial logit: unordered categories, ordinal logit: ordered categories
- Class imbalance is common in economics — address via resampling or class weights
- Next week: Ridge and LASSO — regularized regression