CS 484
Introduction to Machine Learning

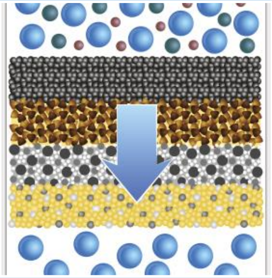
Week 7, March 4, 2021

Spring Semester 2021

ILLINOIS TECH

College of Computing

Week 7: Feature Selection



Filter Method

Wrapper Method

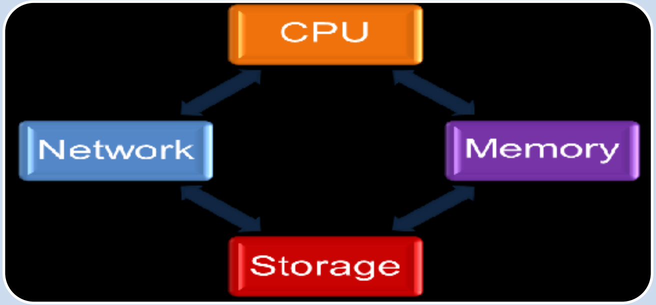
Embedded Method

Feature Selection in The Training Process

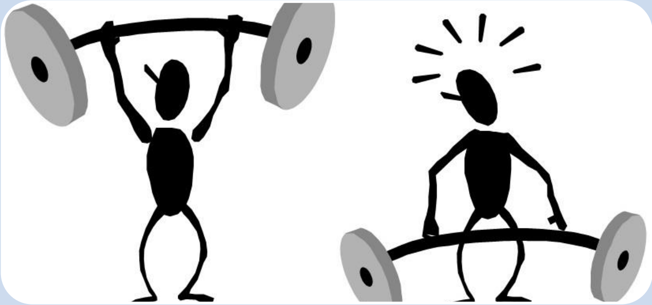
AVOID	• Subjectively Withhold Features From Consideration
IDENTIFY	• Analytically Identify Features That *May* Predict The Target
SELECT	• Let the Algorithm to Select Features into The Model

Why Select Feature in Training Models?

Personal Identifiable Information (PII) or Protected Health Information (PHI) are prohibited from being used for any predictive models



Resources (e.g., memory, disk space, processor speed) limit the number of features that can be fed into a computer program



Weakness in algorithms or shortcomings in implementation (e.g., collinearity detection, missing values handling, and outdated platform restrictions)

Avoid These Features

(At least, seek a second opinion before using them)

PII — Personal Identifiable Information

- Any information that can be used to distinguish or trace an individual's identity such as name, social security number, date and place of birth, mother's maiden name, or biometric records
- Beware of the General Data Protection Regulation (GDPR) in the European Union

PHI — Protected Health Information

- Any information in a medical record that can be used to identify an individual, including billing information, test results, prescription records, communication records, and scheduling records
- Mandated by the Health Insurance Portability and Accountability Act of 1996 (HIPAA)

Discriminatory — Features that challenge the state of fairness

- The Equal Employment Opportunity Act (EEOA) of 1972 and the Age Discrimination in Employment Act (ADEA) of 1967
- EEOA prohibits employment discrimination based on race, color, religion, sex, or national origin
- ADEA forbids age discrimination against people who are age 40 or older

Feature Inclusion Directives

Certain features are included whatsoever due to business practice in an industry

U.S. regions or census tracts for data that have a geographical element

Marketing incentives in studying customer purchase behavior

Use These Features At Your Own Risk

Single Category

- A particular category occurs too often
- Relative frequency of that category is equal to or above a threshold
- Algorithm may not anticipate

Many Distinct Categories

- Many categories occur rarely
- Relative frequencies of the majority of categories are below a threshold
- May consume lot of resources

Constant Value

- The values are within a narrow range
- Coefficient of Variation (Std.Dev. / Mean) is below a threshold
- May cause machine underflow

Missing Value

- Majority of observations are missing
- Percent of missing observations is above a threshold
- Algorithm may not handle

Lack of Variety among Categories

- Suppose a categorical feature has K categories
- Let category j has n_j number of observations, $j = 1, \dots, K$.
- If we can observe the j^{th} category, then it must contain some observations. Therefore, $n_j > 0$.
- Let $\sum_{j=1}^{K} n_j = N > 0$ denotes the total number of observations.
- Let $p_j = n_j / N$ denotes the empirical probability
- The corresponding Entropy value is $E = -\sum_{j=1}^{K} p_j \log_2(p_j)$.

Lack of Variety among Categories

- The maximum Entropy is $E_{max} = \log_2(K)$ corresponds to a boundary scenario A where all $n_j = N/K$.
- Another boundary scenario B where $n_1 = \cdots = n_{K-1} = 1$ and $n_K = N - K + 1$. The Entropy is $E_0 = \log_2(N) - \frac{N-K+1}{N}\log_2(N - K + 1)$.
- Calculate a score that is defined as $(E - E_0)/(E_{max} - E_0)$ that varies between 0 and 1, inclusively. As the general scenario is moving away from scenario B, the score will increase.
- Drop a categorical feature if the score is small.

Seemingly Constant Interval Values

- Suppose the interval feature has the mean $\bar{x}$ and the standard deviation s.
- The Coefficient of Variation $CV = \text{Sign}(\bar{x}) \times \frac{s}{\max(1,|\bar{x}|)}$
- $\text{Sign}(u) = 1$ if $u \geq 0$ and $\text{Sign}(u) = -1$ if $u < 0$
- If the Coefficient of Variation is too small, then the values of the interval feature are likely to be constant.

Feature Screening

```
for thisVar in features:
   thisDType = inputData[thisVar].dtypes

   # Calculate the number and the percent of missing values
   nNaN = inputData[thisVar].isna().sum()
   percentNaN = 100.0 * (nNaN / nRow)

   isString = numpy.NaN
   entropy = numpy.NaN
   percentEntropy = numpy.NaN
   mean = numpy.NaN
   coefVar = numpy.NaN

   # Calculate the number and the percent of unique values
   uniqueValue = inputData[thisVar].value_counts()
   nValid = numpy.sum(uniqueValue)
   uniqueProp = uniqueValue / nValid
   nUnique = uniqueValue.size
   if (nUnique > 0):
      isString = 0
      for i in range(nUnique):
         if (isinstance(uniqueValue.index[i], str) == 1):
            isString = 1
            break
```

Feature Screening

```
    # Calculate the entropy by treating each feature a categorical field
    entropy = - numpy.sum(uniqueProp * numpy.log2(uniqueProp))
    if (nUnique > 1):
       e0 = nValid - nUnique + 1
       e0 = numpy.log2(nValid) - (e0 / nValid) * numpy.log2(e0)
       e1 = numpy.log2(nUnique)
       if (e1 > e0):
          percentEntropy = 100.0 * ((entropy - e0) / (e1 - e0))

    # Calculate the coefficient of variation
    if (isString == 0):
       mean = numpy.mean(inputData[thisVar])
       coefVar = numpy.std(inputData[thisVar], ddof = 1) / max(1.0, abs(mean))
       if (mean < 0.0):
          coefVar = - coefVar

 metaData = metaData.append([[thisVar, thisDType, isString, nValid, nNaN, percentNaN, nUnique, entropy,
                              percentEntropy, mean, coefVar]], ignore_index=True)
```

Feature Screening

```
metaData = metaData.rename(columns = {0: 'Feature Field', 1: 'DType',
                                      2: 'String Value?',  3: 'Number of Valids',
                                      4: 'Number of NaNs', 5: 'Percent of NaNs',
                                      6: 'Number of Unique Values', 7: 'Entropy',
                                      8: 'Percent of Entropy',
                                      9: 'Mean', 10: 'Coefficient of Variation'})
```

Need to Identify Helpful Features

Number of Features

From the viewpoint of a data scientist, too many features may be equally troublesome as too few features

Learner Algorithm

An algorithm is useful and effective only if it can take all the features that are specified, no matter how powerful it is

Identify the Helpful

Therefore, we need to identify the features that help predict the target accurately before specifying them in the algorithm

Identify the Helpful Features

Appropriate Statistical Test		Target Variable	
		Categorical	Interval
Input Feature	Categorical	Chi-square	ANOVA
	Interval	Deviance	Regression

Chi-square Test

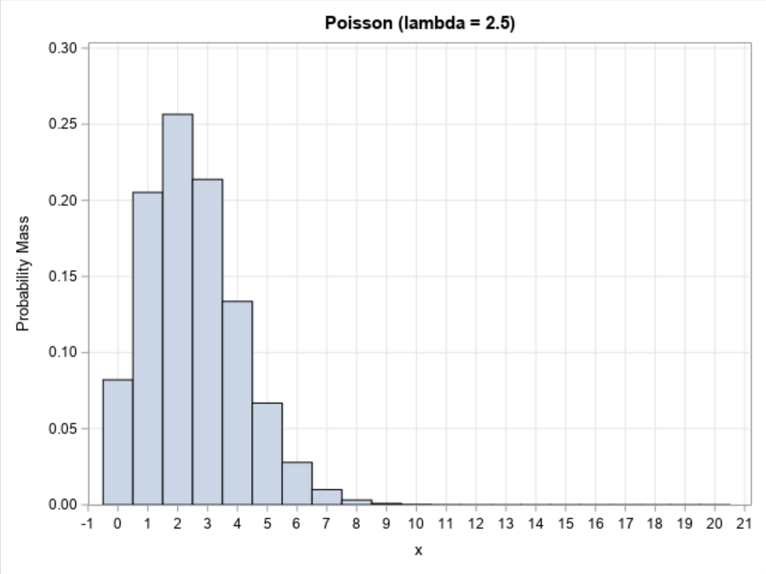
- Suppose the categorical target variable Y has $K > 1$ categories
- Suppose the categorical feature X has $L > 1$ categories
- Let $n_{ij} \geq 0, i = 1, \dots, L$ and $j = 1, \dots, K$ be the number of observations in the i^{th} category of the feature and the j^{th} category of the target.
- The marginal counts are:
  - Across Column: $n_{i+} = \sum_{j=1}^{K} n_{ij}, i = 1, \dots, L$
  - Across Row: $n_{+j} = \sum_{i=1}^{L} n_{ij}, j = 1, \dots, K$
  - Across Cell: $n_{++} = \sum_{i=1}^{L} \sum_{j=1}^{K} n_{ij}$

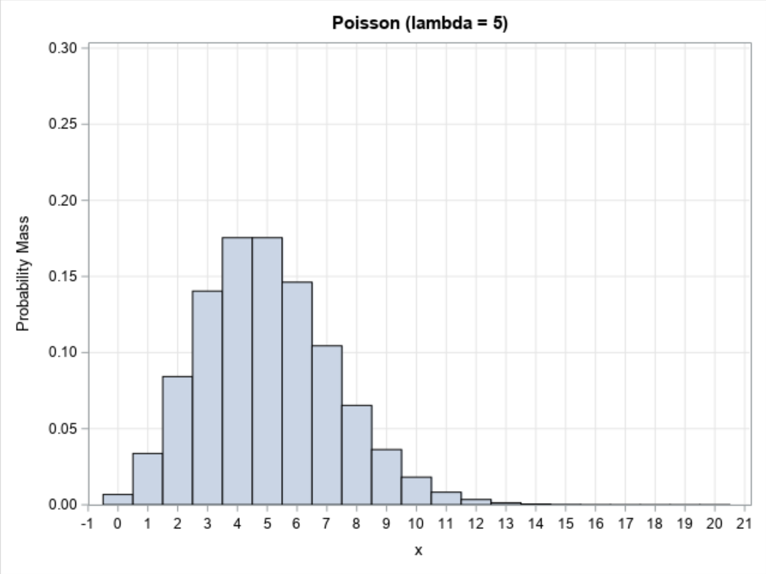
Chi-square Test – Poisson Distribution

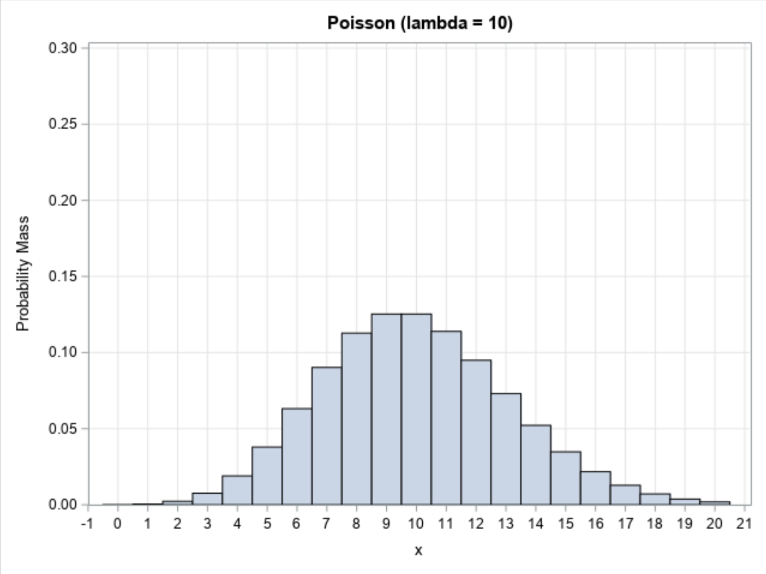
- We assume that the counts $n_{ij} \geq 0$ follow a Poisson distribution which is characterized by a single parameter$\lambda = E_{ij} > 0$.
- The probability mass function is $\Pr(n_{ij} = x) = \frac{\lambda^x}{x!} e^{-\lambda}, x = 0,1, \ldots$
- Under the Poisson distribution,
  - The expectation or the mean of n_{ij} is $E(n_{ij}) = E_{ij}$ and
  - The variance of n_{ij} is $\mathrm{var}(n_{ij}) = E_{ij}$.

Poisson Distribution

Python Function	Density	Distribution	Significance	Quantile	Random Number
scipy.stats.poisson	.pmf()	.cdf()	.sf()	.ppf()	.rvs()







Chi-square Test

- The normalized term $\frac{n_{ij}-E_{ij}}{\sqrt{E_{ij}}}$ asymptotically follows a standard normal distribution when n_{ij} is large.
- In other words, $\frac{\left(n_{ij}-E_{ij}\right)^2}{E_{ij}}$ asymptotically follows a Chi-square distribution with one degree of freedom.
- What is the distribution of $\sum_{i=1}^{L}\sum_{j=1}^{K}\frac{\left(n_{ij}-E_{ij}\right)^2}{E_{ij}}$?

Chi-square Test

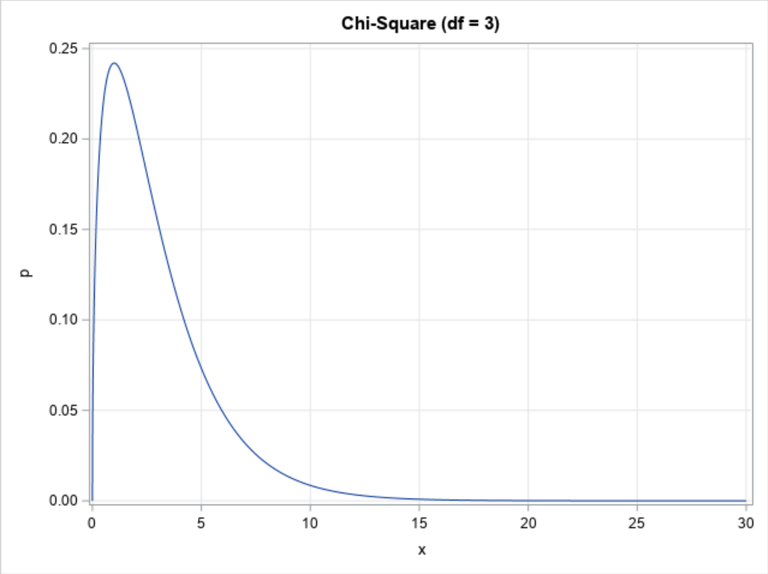
- Karl Pearson (1857 – 1936), an English mathematician and biostatistician, proved that $\sum_{i=1}^{L}\sum_{j=1}^{K}\frac{\left(n_{ij}-E_{ij}\right)^2}{E_{ij}}$ follows a Chi-Square distribution with positive degrees of freedom asymptotically.
- The degrees of freedom depends on how the E_{ij} are computed.

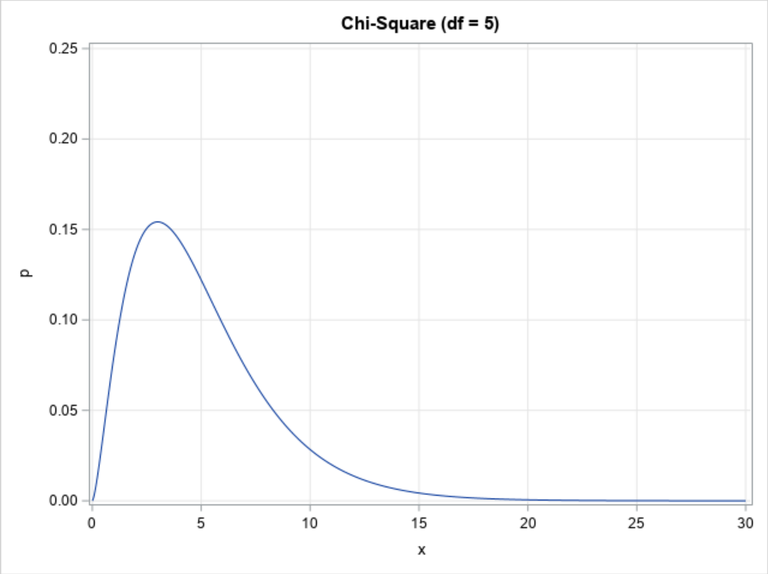
Chi-Square Distribution

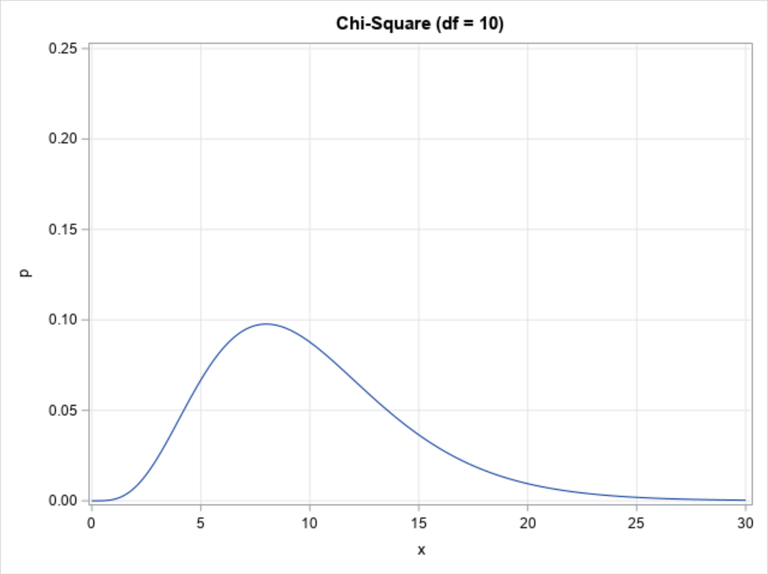
- Let V be a continuous random variable that takes values from this open interval $(0, \infty)$.
- The probability density function is $f(v) = \frac{1}{2^{m/2}\Gamma(m/2)} v^{\frac{m}{2}-1} e^{-\frac{v}{2}}$
  - $m > 0$ is the degrees of freedom
  - $\Gamma(u)$ is the complete Gamma function
- The probability $\Pr(V < v) = \int_0^v f(u)\, du$ (i.e., no closed-form).
- The mean is m and the variance is $2m$.
- When $m = 2$, the Chi-square distribution is an Exponential distribution with $\lambda = 1/2$.

Chi-Square Distribution

Python Function	Density	Distribution	Significance	Quantile	Random Number
scipy.stats.chi2	.pdf()	.cdf()	.sf()	.ppf()	.rvs()







Chi-square Test – Expected Count

Feature	Target		
	No	**Yes**	**Total**
A	4	6	10
B	2	3	5
C	8	12	20
Total	**14**	**21**	**35**

IF the feature is statistically independent of the target variable, then the feature will not help predict the target variable

THEN the relative frequencies (or percentages) across the target categories are the same for each feature's category

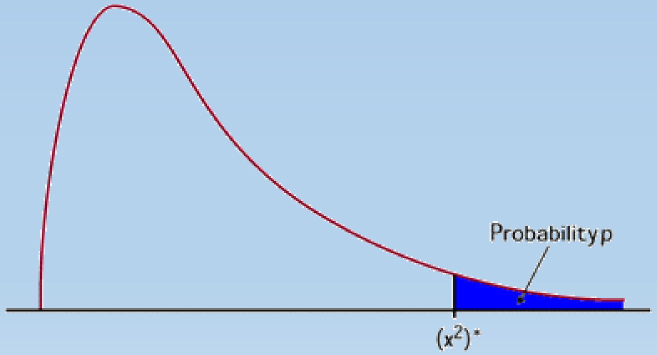
THEREFORE, the common relative frequencies are be estimated by $n_{+j}/n_{++}, j = 1, \dots, K$

HENCE, the expected count in the (i, j) cell is $E_{ij} = n_{i+}\dfrac{n_{+j}}{n_{++}}$

Feature	Target		
	No	**Yes**	**Total**
A	0.4	0.6	1
B	0.4	0.6	1
C	0.4	0.6	1
Total	**0.4**	**0.6**	**1**

Chi-square Test - Procedure

1. The expected count in the (i, j) cell is $E_{ij} = n_{i+}\frac{n_{+j}}{n_{++}} > 0$
2. The Pearson Chi-square statistic is $\chi^2 = \sum_{i=1}^{L}\sum_{j=1}^{K}\frac{\left(n_{ij}-E_{ij}\right)^2}{E_{ij}}$
3. The one-sided significance (the probability on the right tail) from the Chi-square distribution with $(L-1)(K-1)$ degrees of freedom
4. Reject the Independence Assumption if the significance value is less than α, say 0.05, and identify this feature as useful



Chi-square Test - Procedure

```
# Define a function that performs the Pearson Chi-square test
#   xCat - Input categorical feature (array-like or Series)
#   yCat - Input categorical target field (array-like or Series)

def PearsonChiSquareTest (xCat, yCat):
    # Generate the crosstabulation
    obsCount = pandas.crosstab(index = xCat, columns = yCat, margins = False, dropna = True)
    xNCat = obsCount.shape[0]
    yNCat = obsCount.shape[1]
    cTotal = obsCount.sum(axis = 1)
    rTotal = obsCount.sum(axis = 0)
    nTotal = numpy.sum(rTotal)
    expCount = numpy.outer(cTotal, (rTotal / nTotal))

    # Calculate the Chi-Square statistics
    chiSqStat = ((obsCount - expCount)**2 / expCount).to_numpy().sum()
    chiSqDf = (xNCat - 1) * (yNCat - 1)
    if (chiSqDf > 0):
       chiSqSig = sdist.chi2.sf(chiSqStat, chiSqDf)
    else:
       chiSqSig = numpy.NaN

    return (xNCat, yNCat, chiSqStat, chiSqDf, chiSqSig)
```

Analysis of Variance (ANOVA) Test

- Suppose the categorical feature X has $L > 1$ categories
- Suppose the interval target is Y.
- Let $y_{ij}, j = 1, \ldots, n_i$ and $i = 1, \ldots, L$ be the observations in the i^{th} category of the feature.
- Let $\bar{y}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} y_{ij}$ be the target's mean in the i^{th} category of the feature.
- Let $\bar{y} = \frac{\sum_{i=1}^{L} \sum_{j=1}^{n_i} y_{ij}}{\sum_{i=1}^{L} n_i}$ be the target's overall mean.

Analysis of Variance (ANOVA) Test

- Within-Group
  - Sum of Squares $SSW = \sum_{i=1}^{L} \sum_{j=1}^{n_i} \left(y_{ij} - \bar{y}_i\right)^2$
  - Degrees of Freedom $DFW = \sum_{i=1}^{L} (n_i - 1)$.
- Total
  - Sum of Squares $SST = \sum_{i=1}^{L} \sum_{j=1}^{n_i} \left(y_{ij} - \bar{y}\right)^2$
  - Degrees of Freedom $DFT = \sum_{i=1}^{L} n_i - 1$.
- Between-Group
  - Sum of Squares $SSG = SST - SSW$
  - Degrees of Freedom $DFG = DFT - DFW = L - 1$.

Analysis of Variance (ANOVA) Test

IF the feature is statistically independent of the target variable, then the feature will not help predict the target variable

THEN the target has the same mean across all feature's categories

THEREFORE, both SSG/DFG and SSW/DFW both estimate the population variance.

HENCE, the ratio $F = \frac{SSG/DFG}{SSW/DFW}$ follows a F distribution

Continuous – F Distribution

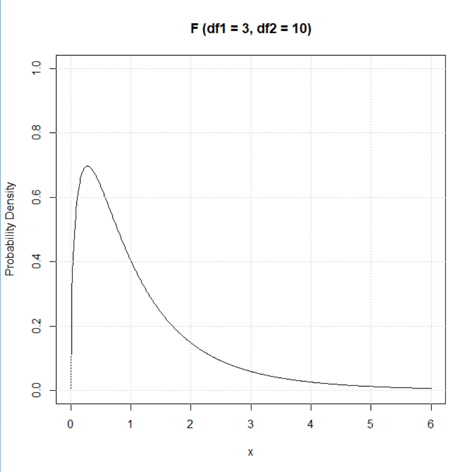
- The probability density function is

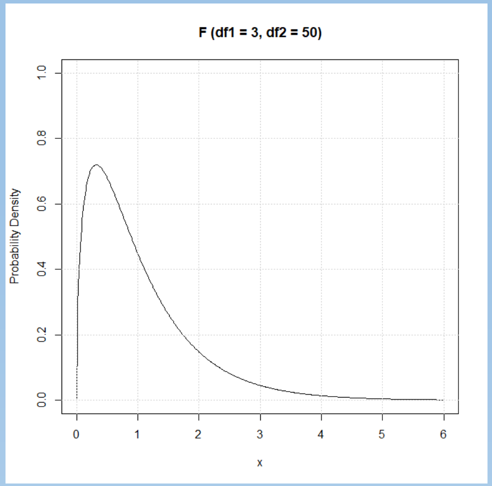
$$f(x) = \frac{\Gamma((\nu_1 + \nu_2)/2)}{\Gamma(\nu_1/2)\Gamma(\nu_2/2)} \left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_1}{2}} x^{\frac{\nu_1}{2}-1} \left(1 + \frac{\nu_1}{\nu_2} x\right)^{-\frac{\nu_1+\nu_2}{2}}$$

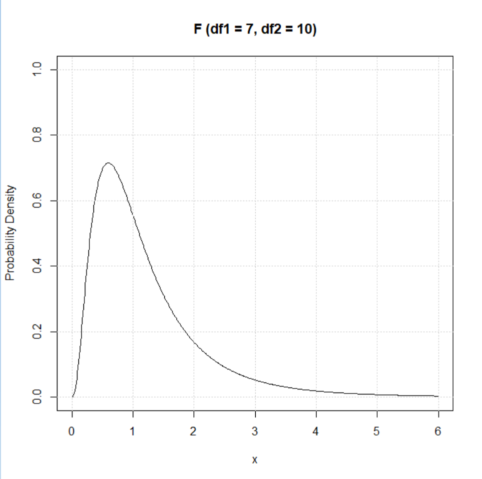
- The domain is $0 < x < \infty$.
- $\nu_1 > 0$ is the first (numerator) degrees of freedom
- $\nu_2 > 0$ is the second (denominator) degrees of freedom
- $\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} dt, a > 0$ is the complete Gamma function
- The mean is $\frac{\nu_2}{(\nu_2-2)}$ and the variance is $\frac{2\nu_2^2(\nu_1+\nu_2-2)}{\nu_1(\nu_2-2)^2(\nu_2-4)}$.

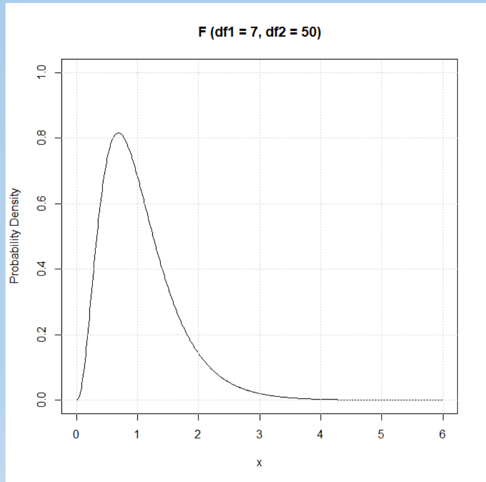
Continuous – F Distribution

Python Function	scipy.stats.f
Density	.pdf()
Distribution	.cdf()
Significance	.sf()
Quantile	.ppf()
Random Number	.rvs()



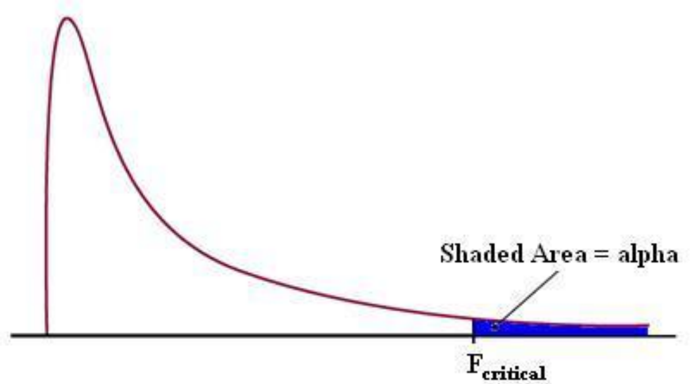






F Test - Procedure

1. The SSG, the DFG, the SSW, and the DFW
2. The statistic $F = \frac{SSG/DFG}{SSW/DFW}$
3. The one-sided significance (the probability on the right tail) from the F distribution with dfn = DFG and dfd = DFW degrees of freedom
4. Reject the Independence Assumption if the significance value is less than α, say 0.05, and identify this feature as useful



F Test - Procedure

```
# Define a function that performs the ANOVA test
#   xCat - Input categorical feature (array-like or Series)
#   yCont - Input continuous target field (array-like or
Series)

def AnalysisOfVarianceTest (xCat, yCont):
   df = pandas.DataFrame(columns = ['x', 'y'])
   df['x'] = xCat
   df['y'] = yCont

   # Total Count and Sum of Squares
   totalCount = df['y'].count()
   totalSSQ = df['y'].var(ddof = 0) * totalCount

   # Within Group Count and Sums of Squares
   groupCount = df.groupby('x').count()
   groupSSQ = df.groupby('x').var(ddof = 0) * groupCount
   nGroup = groupCount.shape[0]

   withinSSQ = numpy.sum(groupSSQ.values)
   betweenSSQ = max(0.0, (totalSSQ - withinSSQ))

   # Compute F statistics
   fDf1 = (nGroup - 1)
   fDf2 = (totalCount - nGroup)
   if (fDf1 > 0 and fDf2 > 0 and withinSSQ > 0.0):
      fStat = (betweenSSQ / fDf1) / (withinSSQ / fDf2)
      fSig = sdist.f.sf(fStat, fDf1, fDf2)
   else:
      fStat = numpy.NaN
      fSig = numpy.NaN

   xNCat = nGroup
   return (xNCat, fStat, fDf1, fDf2, fSig)
```

Regression Test

- Suppose $(x_i, y_i), i = 1, \ldots, n$ are the pairs of values for the feature X and the target variable Y.
- Let $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ be the mean of the feature
- Let $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ be the mean of the target variable
- If we fit this regression line $y = \alpha + \beta x$, the regression coefficient is $\hat{\beta} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$ and the intercept is $\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$.
- The fitted values are $\hat{y}_i = \hat{\alpha} + \hat{\beta} x_i, i = 1, \ldots, n$.

Regression Test

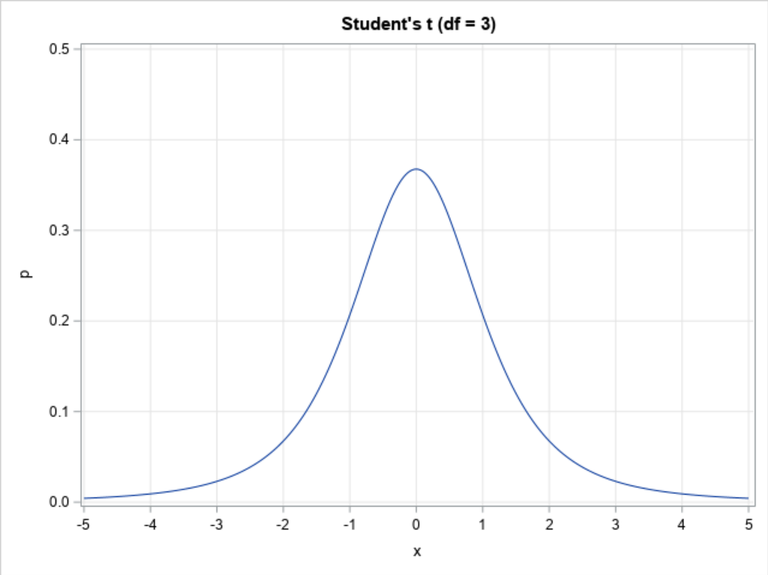
- The residual sum of squares is $RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.
- The estimated residual variance is $\widehat{\sigma^2} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-2}$
- The standard error of $\hat{\beta}$ is $\mathrm{se}(\hat{\beta}) = \sqrt{\frac{\widehat{\sigma^2}}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}$.
- The statistic $t = \frac{\widehat{\beta}}{\mathrm{se}(\widehat{\beta})}$ follows a Student's *t* distribution with $n - 2$ degrees of freedom.

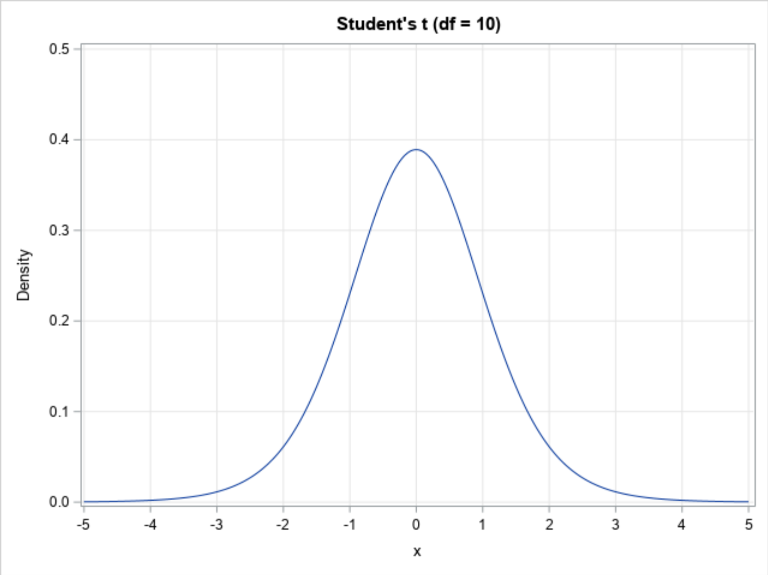
Continuous – Student's *t* Distribution

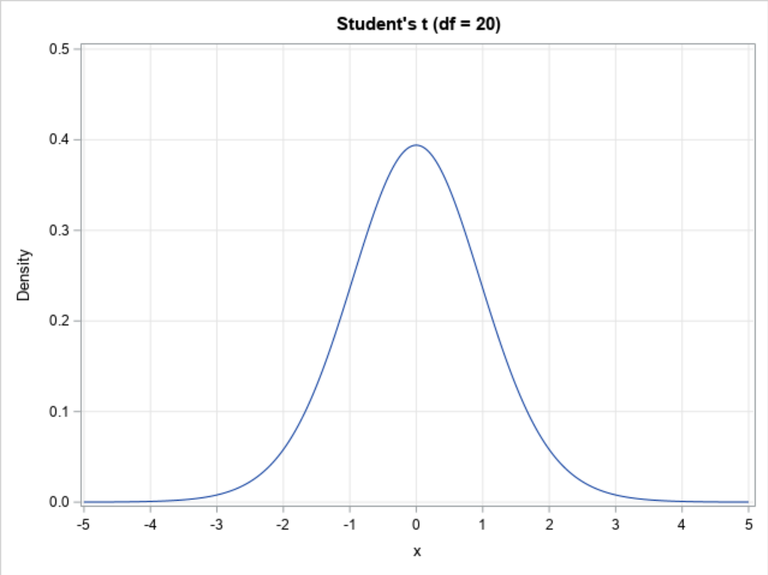
- Let X be a continuous random variable that takes values from this open interval $(-\infty, \infty)$.
- The probability density function is $f(x) = \dfrac{\Gamma\left(\frac{k+1}{2}\right)}{\sqrt{k\pi}\,\Gamma\left(\frac{k}{2}\right)}\left(1+\dfrac{x^2}{k}\right)^{-\frac{k+1}{2}}$
  - $k > 0$ is the degrees of freedom
  - $\Gamma(u)$ is the complete Gamma function
- The probability $\Pr(X < x) = \int_0^x f(u)\, du$ (i.e., no closed-form).
- The mean is 0 and the variance is $k/(k-2)$.
  - The variance does not exist if $k \leq 2$.

Continuous – Student’s *t* Distribution

Python Function	Density	Distribution	Significance	Quantile	Random Number
scipy.stats.t	.pdf()	.cdf()	.sf()	.ppf()	.rvs()







Regression Test

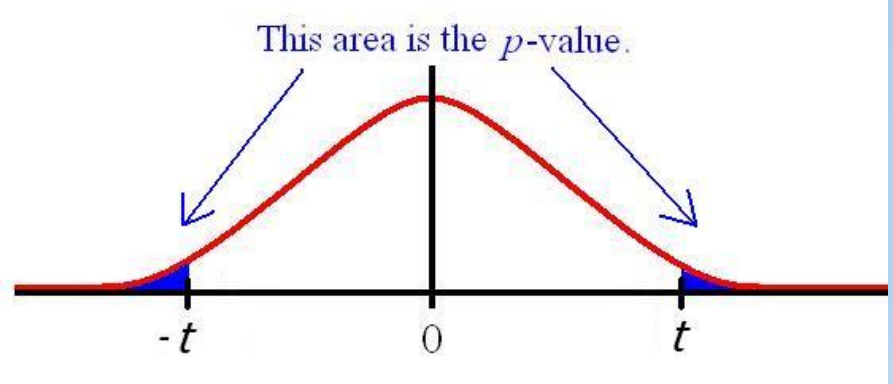
IF the feature is statistically independent of the target variable, then the feature will not help predict the target variable

THEN the regression coefficient β is zero

THEREFORE, the absolute value of statistic $t = \frac{\hat{\beta}}{\mathrm{se}(\hat{\beta})}$ should be small.

Student's *t* Test - Procedure

1. The regression coefficient $\hat{\beta}$ and its standard error $\mathrm{se}\left(\hat{\beta}\right)$
2. The statistic $t = \hat{\beta}/\mathrm{se}\left(\hat{\beta}\right)$
3. The two-sided significance (the probability on both tails) from the Student's *t* distribution with $n - 2$ degrees of freedom
4. Reject the Independence Assumption if the significance value is less than α, say 0.05, and identify this feature as useful



Student's *t* Test - Procedure

```
# Define a function that performs the Regression test
#   xCont - Input continuous feature (array-like or Series)
#   yCont - Input continuous target field (array-like or Series)

def RegressionTest (xCont, yCont):
   nObs = len(yCont)
   xyCov = numpy.cov(xCont, yCont, ddof = 0)
   tDf = nObs - 2
   tStat = numpy.NaN
   tSig = numpy.NaN
   if (tDf > 0 and xyCov[0,0] > 0.0):
      xMean = numpy.mean(xCont)
      yMean = numpy.mean(yCont)
      regB = xyCov[0,1] / xyCov[0,0]
      yHat = yMean + regB * (xCont - xMean)
      residVariance = numpy.sum((yCont - yHat)**2) / tDf
      if (residVariance > 0.0):
         seB = numpy.sqrt(residVariance / (nObs * xyCov[0,0]))
         tStat = regB / seB
         tSig = 2.0 * sdist.t.sf(abs(tStat), tDf)

   return (tStat, tDf, tSig)
```

Deviance Test

- Suppose the interval feature is X
- Suppose the categorical target variable Y has $K > 1$ categories

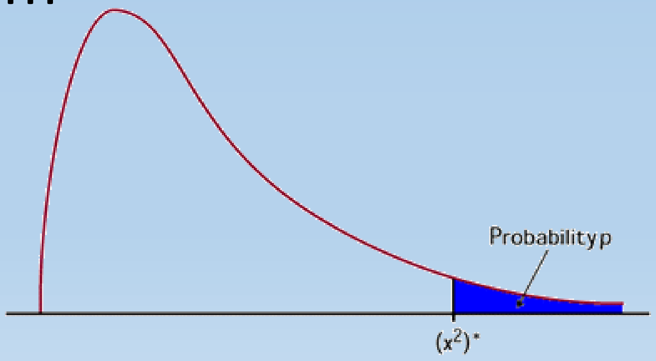
IF the feature is statistically independent of the target variable, then the feature will not help predict the target variable

THEN the goodness-of-fit of a logistic model with the interval feature X should be the same as another logistic model without the interval feature is X

THEREFORE, the Deviance statistic should be small

Deviance Test - Procedure

1. Build a multinomial logistic model with only the Intercept term. Let l_0 be the model log-likelihood value.
2. Build another multinomial logistic model with the Intercept term and the interval feature X. Let l_1 be the model log-likelihood value.
3. Calculate this statistic $G^2 = 2(l_1 - l_0)$
4. The one-sided significance (the probability on right tail) from the Chi-square distribution with $K - 1$ degrees of freedom
5. Reject the Independence Assumption if the significance value is less than α, say 0.05, and identify this feature as useful



Deviance Test - Procedure

```
# Define a function that performs the Deviance Chi-square test
#   xCont - Input categorical feature (array-like or Series)
#   yCat - Input categorical target field (array-like or Series)

def DevianceTest (xCont, yCat):
    # Train a model with the Intercept term and xCont
    X = smodel.add_constant(xCont, prepend = True)
    y = yCat.astype('category')
    logit = smodel.MNLogit(y, X)
    yNCat = logit.J
    thisFit = logit.fit(method = 'newton', maxiter = 1000, full_output = False, disp = False)
    chiSqStat = thisFit.llr
    chiSqDf = thisFit.df_model
    chiSqSig = thisFit.llr_pvalue

    return (yNCat, chiSqStat, chiSqDf, chiSqSig)
```

The Home Equity Loan Example

Categorical Feature: REASON, JOB, DEROG, DELINQ, and NINQ

Interval Feature: LOAN, MORTDUE, VALUE, YOJ, CLAGE, CLNO, and DEBTINC

5,960 records, 13 variables read from hmeq.csv

3,364 complete records analyzed

Categorical Target Variable: BAD

Chi-square Test for categorical features

Deviance Test for Interval Feature

Ranking based on Significance

The Home Equity Loan Example

Week 7 Identify Useful Feature for Cat Target.py

```
# The Home Equity Loan example
catPred = ['REASON', 'JOB', 'DEROG', 'DELINQ', 'NINQ']
intPred = ['LOAN', 'MORTDUE', 'VALUE', 'YOJ', 'CLAGE', 'CLNO', 'DEBTINC']

hmeq = pandas.read_csv('C:\\IIT\\Machine Learning\\Data\\hmeq.csv',
                       delimiter=',', usecols = ['BAD']+catPred+intPred)
hmeq = hmeq.dropna()

testResult = pandas.DataFrame()

for pred in catPred:
    xNCat, yNCat, chiSqStat, chiSqDf, chiSqSig = PearsonChiSquareTest(hmeq[pred], hmeq['BAD'])
    testResult = testResult.append([[pred, xNCat, yNCat, chiSqStat, chiSqDf, chiSqSig]], ignore_index = True)

for pred in intPred:
    yNCat, chiSqStat, chiSqDf, chiSqSig = DevianceTest(hmeq[pred], hmeq['BAD'])
    testResult = testResult.append([[pred, numpy.NaN, yNCat, chiSqStat, chiSqDf, chiSqSig]], ignore_index = True)

testResult = testResult.rename(columns = {0:’Feature', 1: 'Feature N Category', 2: 'Target N Category',
                                          3:'Statistic', 4:'DF', 5:'Significance'})
rankSig = testResult.sort_values('Significance', axis = 0, ascending = True)
```

The Home Equity Loan Example

Type	Feature	Feature N Category	Target N Category	Statistic	DF	Significance
Categorical	REASON	2	2	0.1313	1	0.7171
	JOB	6	2	36.2547	5	8.4465E-07
	DEROG	11	2	237.8857	10	1.9039E-45
	DELINQ	10	2	302.7278	9	6.8868E-60
	NINQ	13	2	97.5806	12	1.6558E-15
Interval	LOAN		2	3.5111	1	0.0610
	MORTDUE		2	0.9512	1	0.3294
	VALUE		2	2.4398	1	0.1183
	YOJ		2	14.8204	1	0.0001
	CLAGE		2	50.6898	1	1.0818E-12
	CLNO		2	0.1896	1	0.6632
	DEBTINC		2	144.4416	1	2.8447E-33

The Home Equity Loan Example

Using the tolerance level $\alpha = 0.05$, seven features (shown on the right) are identified as helpful in predicting the categorical target BAD

Categorical	Interval
• DELINQ	• DEBTINC
• DEROG	• CLAGE
• NINQ	• YOJ
• JOB	

Feature Importance

- In the previous example, the significances of seven features are less than 0.05.
- Among them:
  - DELINQ has the smallest significance at 6.89E-60
  - YOJ has the largest significance at 0.0001
- Do these significances carry the same weights in our decision?
- Do these significances indicate that some features are more important than others?

Common Measures for Feature Importance

Significance — Rank significances on the logarithm scale

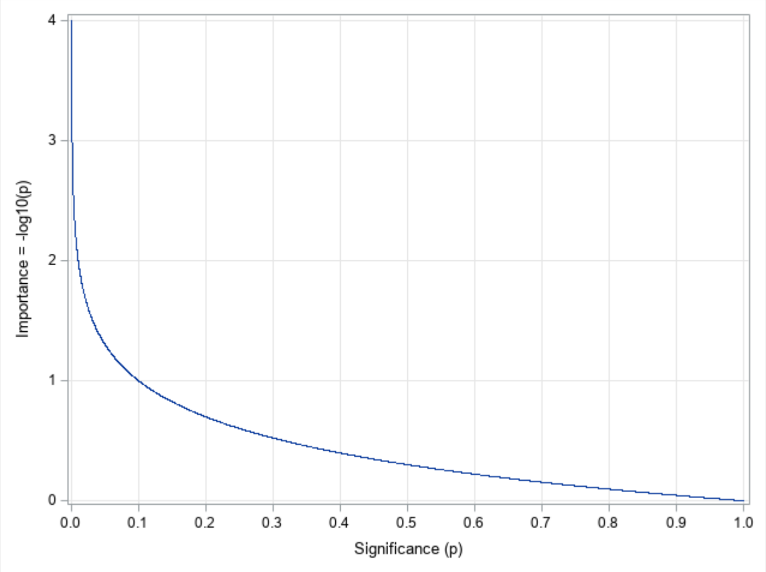
- Suppose the significance of a statistical test is $0 < p \leq 1$, then calculate importance as $-\log_{10}(p)$.
- The higher the importance value, the more predictive a feature for the target variable.

Association — Measure the strength of association

- Calculate the appropriate statistic for measuring the strength of association between the feature and the target variable.

Feature Importance: $-\log_{10}(\mathbf{Significance})$

- Suppose the significance of a statistical test is $0 < p \leq 1$, then calculate importance as $-\log_{10}(p)$.
  - If $p = 1$, then the importance measure is zero
  - If p is zero because the significance value falls below the machine precision, then we define the importance measure as infinity
- The higher the importance value, the more predictive a feature for the target variable.



Note: The machine precision is 2^{-52} for the 64-bit floating point arithmetic

Python: numpy.finfo(float).eps

The Home Equity Loan Example

Feature	Feature N Category	Target N Category	Statistic	DF	Significance	-log10(Significance)
DELINQ	10	2	302.7278	9	6.8868E-60	59.2
DEROG	11	2	237.8857	10	1.9039E-45	44.7
DEBTINC		2	144.4416	1	2.8447E-33	32.5
NINQ	13	2	97.5806	12	1.6558E-15	14.8
CLAGE		2	50.6898	1	1.0818E-12	12.0
JOB	6	2	36.2547	5	8.4465E-07	6.1
YOJ		2	14.8204	1	0.0001	3.9
LOAN		2	3.5111	1	0.0610	1.2
VALUE		2	2.4398	1	0.1183	0.9
MORTDUE		2	0.9512	1	0.3294	0.5
CLNO		2	0.1896	1	0.6632	0.2
REASON	2	2	0.1313	1	0.7171	0.1

Week 7 Identify Useful Feature for Cat Target.py

Association Based Feature Importance

- We prefer the association-based importance measure to be:
  - Positive
  - Between zero and one
  - A higher value indicates more important, and a lower value indicates less important

Appropriate Association Measure		Target Variable	
		Categorical	Interval
Input Feature	Categorical	Cramer's V	Eta-squared
	Interval	McFadden's Pseudo R^2	Squared Pearson Correlation

Cramer's V

- Suppose the categorical target variable Y has $K > 1$ categories
- Suppose the categorical feature X has $L > 1$ categories
- Let $n_{ij} \geq 0, i = 1, \ldots, L$ and $j = 1, \ldots, K$ be the number of observations in the i^{th} category of the feature and the j^{th} category of the target.
- The marginal counts are:
  - Across Column: $n_{i+} = \sum_{j=1}^{K} n_{ij}, i = 1, \ldots, L$
  - Across Row: $n_{+j} = \sum_{i=1}^{L} n_{ij}, j = 1, \ldots, K$
  - Across Cell: $n_{++} = \sum_{i=1}^{L} \sum_{j=1}^{K} n_{ij}$

Cramer's V

- The expected count in the (i, j) cell is $E_{ij} = n_{i+} \frac{n_{+j}}{n_{++}} > 0$
- The Pearson Chi-square statistic is $\chi^2 = \sum_{i=1}^{L} \sum_{j=1}^{K} \frac{\left(n_{ij} - E_{ij}\right)^2}{E_{ij}}$
- The Cramer's V is $V = \sqrt{\frac{\chi^2}{n_{++} \times \min(K-1, L-1)}}$
- The Cramer's V is between 0 and 1
- A larger Cramer's V indicates that a particular category in the feature is associated with a particular category in the target variable

Eta-squared

- Within-Group Sum of Squares $SSW = \sum_{i=1}^{L} \sum_{j=1}^{n_i} \left(y_{ij} - \bar{y}_i\right)^2$
- Total Sum of Squares $SST = \sum_{i=1}^{L} \sum_{j=1}^{n_i} \left(y_{ij} - \bar{y}\right)^2$
- Between-Group Sum of Squares $SSG = SST - SSW$
- The Eta-squared is $\eta^2 = \frac{SSG}{SST}$
- The Eta-squared is between 0 and 1
- A larger η^2 indicates that the means of the target variable in the feature's categories are different.

Squared Pearson Correlation

- Suppose $(x_i, y_i), i = 1, \dots, n$ are the pairs of values for the feature X and the target variable Y.
  - Let $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ be the mean of the feature
  - Let $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ be the mean of the target variable
- The squared Pearson correlation is $r^2 = \frac{\left(\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})\right)^2}{\left(\sum_{i=1}^{n}(x_i - \bar{x})^2\right)\left(\sum_{i=1}^{n}(y_i - \bar{y})^2\right)}$
- The squared Pearson correlation is between 0 and 1
- A larger r^2 indicates that the larger-than-mean feature values are associated with larger (or smaller)-than-mean target values

McFadden's Pseudo R^2

- Build a multinomial logistic model with only the Intercept term. Let l_0 be the model log-likelihood value.
- Build another multinomial logistic model with the Intercept term and the interval feature X. Let l_1 be the model log-likelihood value.
- The McFadden's Pseudo R-squared is $R^2_{\text{McF}} = 1 - \frac{l_1}{l_0}$
- The McFadden's Pseudo R-squared is between 0 and 1
- If the interval feature does not help predict the categorical target, then $l_1 = l_0$ and $R^2_{\text{McF}} = 0$.
- If the interval feature completely predicts the categorical target (i.e., complete separation), then $l_1 = 0$ and $R^2_{\text{McF}} = 1$.

Remarks on Feature Importance

Significance-based metric depends on the statistical power of the test in rejecting the Independence Assumption

Association-based metric may have different interpretation between categorical and interval features

Wrapper Method

- This method selects features by using the model assessment metrics, e.g., sum of squared residuals, or log-likelihood value
- Determine a subset of features that generates the optimal metrics.
  - Exhaustive Search – All Possible Subset
  - Directional Search – Forward / Backward Selection
  - Random Search – Generate a Random Subset of Features
- Consider using the AIC or the BIC values to balance our desire for optimal metrics and the need for a model with fewer parameters.

Embedded Method

- The Embedded method is built into a machine learning algorithm.
- There are primarily two approaches in the Embedded method.
  1. The first approach manipulates the model parameter estimates to either boost or diminish a feature's influence on the predicted target values.
  2. The second approach provides us feature importance indices.

First Approach of Embedded Method

- The first approach manipulates the model parameter estimates to either boost or diminish a feature’s influence on the predicted target values.
- This approach is available only to machine learning algorithms that train parametric models.
- The Regularization method is a common representative of the first approach.

Second Approach of Embedded Method

- The second approach lets a machine learning algorithm runs its course, but it will train the model under simulated scenarios.
- This approach collects model information under simulated sceanios to evaluate the effectiveness of a feature in predicting the model outcomes.
- The evaluation results are then compiled to give us the feature importance indices.
- The tree-based random forest method is a common representative of the second approach.

In the Interest of Time …

- In the interest of time, we will focus only on the Regularization method.
- Besides, we are more interested in training a model with just the *right* number of features than knowing their relative importance.

Regularization Method

- Denote the parameters as $\boldsymbol{\beta}^t = (\beta_1, \dots, \beta_p)$, the objective function as $l(\boldsymbol{\beta})$, and the penalty term as $T(\boldsymbol{\beta}) \geq 0$.
- The Regularization method will minimize $C \times l(\boldsymbol{\beta}) + T(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta} \in \boldsymbol{\Omega}$ where $\boldsymbol{\Omega}$ is a domain space for the parameters. Often, $\boldsymbol{\Omega} = \mathbb{R}^p$ the p-dimension real number space.
- The $C > 0$ multiplier specifies the relative regularization strength.
  - Values less than one, i.e., $0 < C < 1$, specify stronger regularization.
  - Values greater than one, i.e., $C > 1$ specify weaker regularization.
  - When $C = 1$, then both the objective function and the penalty term contribute equally to the optimization.

L1 Regularization

- Penalty term is $T(\boldsymbol{\beta}) = \sum_{j=1}^{p} \left|\beta_j\right|$.
- Also known as LASSO Regression (LASSO stands for Least Absolute Shrinkage and Selection Operator).
- This penalty is applied linearly to all magnitudes of parameters.
- The penalty will force the parameters of the unselected features down to zero but tolerate the selected feature to have justifiable larger parameter values.

L2 Regularization

- Penalty term is $T(\boldsymbol{\beta}) = \sum_{j=1}^{p} \beta_j^2$.
- Also known as Ridge Regression.
- This penalty aggressively persuades all features to avoid large parameter values, period. Otherwise, the penalty is quadratically increases.
- On the contrary, it does not force the unselected features' parameters to zero only if these features keep their parameter values at bay (i.e., small enough without increasing the penalty substantially).

L1/L2 Regularization

- Penalty Term is $T(\boldsymbol{\beta}) = \phi \sum_{j=1}^{p} |\beta_j| + (1 - \phi) \sum_{j=1}^{p} \beta_j^2$ where $0 \leq \phi \leq 1$.
- Also known as Elastic Nets.
- This penalty is a combination of the L1 Regularization and the L2 Regularization.
- We can adjust the ϕ value to take advantage of the merits of both regularizations and to downplay their drawbacks.

Regularization on Logistic Regression

```
import sklearn.linear_model as linear_model
import sklearn.metrics as metrics
import statsmodels.api as smodel

# The Home Equity Loan example
catTarget = 'BAD'
intPred = ['LOAN', 'MORTDUE', 'VALUE', 'YOJ', 'CLAGE', 'CLNO', 'DEBTINC']

hmeq = pandas.read_csv('C:\\IIT\\Machine Learning\\Data\\hmeq.csv',
                       delimiter=',', usecols = [catTarget]+intPred)
trainData = hmeq.dropna()

Y = trainData[catTarget].astype('category')
fullX = trainData[intPred]
fullX.insert(0, '_Intercept', 1.0)

XtX = numpy.transpose(fullX).dot(fullX)             # The SSCP matrix
pDim = XtX.shape[0]

invXtX, aliasParam, nonAliasParam = SWEEPOperator(pDim, XtX, 1.0e-8)
print(fullX.columns[list(aliasParam)])

modelX = fullX.iloc[:, list(nonAliasParam)].drop('_Intercept', axis = 1)
```

Regularization on Logistic Regression

```
# Logistic regression with L1 regularization
objLogit = linear_model.LogisticRegression(penalty = 'l1', fit_intercept = True, random_state = 31008,
                                           solver = 'liblinear', max_iter = 1000, tol = 1e-4)
thisFit = objLogit.fit(modelX, Y)
intercept_l1 = thisFit.intercept_
param_l1 = pandas.Series(thisFit.coef_[0,:], index = modelX.columns)
predProb = objLogit.predict_proba(modelX)
AUC_l1 = metrics.roc_auc_score(Y, predProb[:,1])

# Logistic regression with L2 regularization
objLogit = linear_model.LogisticRegression(penalty = 'l2', fit_intercept = True,  random_state = 31008,
                                           max_iter = 1000, tol = 1e-4)
thisFit = objLogit.fit(modelX, Y)
intercept_l2 = thisFit.intercept_
param_l2 = pandas.Series(thisFit.coef_[0,:], index = modelX.columns)
predProb = objLogit.predict_proba(modelX)
AUC_l2 = metrics.roc_auc_score(Y, predProb[:,1])
```

Regularization on Logistic Regression

```
# Logistic regression with L1/L2 regularization
objLogit = linear_model.LogisticRegression(penalty = 'elasticnet', fit_intercept = True,
                                           l1_ratio = 0.5, solver = 'saga', max_iter = 10000,
                                           tol = 1e-4,  random_state = 31008)
thisFit = objLogit.fit(modelX, Y)
intercept_elasticnet = thisFit.intercept_
param_elasticnet = pandas.Series(thisFit.coef_[0,:], index = modelX.columns)
predProb = objLogit.predict_proba(modelX)
AUC_elasticnet = metrics.roc_auc_score(Y, predProb[:,1])

# Logistic regression without any regularization
modelX = smodel.add_constant(modelX, prepend = True)
objLogit = smodel.MNLogit(Y, modelX)
thisFit = objLogit.fit(method = 'ncg', maxiter = 200, tol = 1e-8)
param_none = thisFit.params
pvalue_none = thisFit.pvalues
predProb = thisFit.predict(modelX)
AUC_none = metrics.roc_auc_score(Y, predProb[1])
```

Regularization on Logistic Regression

Regularization	Area Under Curve
None	0.5405222
L1 (LASSO)	0.7192331
L2 (Ridge)	0.6362205
L1/L2 (Elastic Nets, ϕ = 0.5)	0.5581801

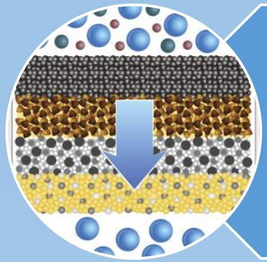
Does constraining the sizes of the parameter estimates really produce a better model?

Regularization on Logistic Regression

Parameter	None	L1	L2	L1/L2
const	-0.0000000001	-2.7397325	-0.0000377	-0.0000037
LOAN	-0.0000014545	-0.0000192	-0.0000237	-0.0000515
MORTDUE	-0.0000057393	-0.0000068	-0.0000003	-0.0000025
VALUE	-0.0000081133	0.0000050	-0.0000025	-0.0000102
YOJ	-0.0000000007	-0.0132115	-0.0005360	-0.0000525
CLAGE	-0.0000000142	-0.0063423	-0.0099510	-0.0009539
CLNO	-0.0000000017	0.0063377	-0.0005577	-0.0000539
DEBTINC	-0.0000000025	0.1018101	-0.0002309	-0.0000207

Does L1 Regularization make the parameter estimates for LOAN, MORTDUE, and VALUE zeros relative to others?

Lecture Recap



Filter Method

- Use business rules or statistical tests
- Easy to implement and understand

Wrapper Method

- Work well with supervised machine learning algorithms
- Exhaustive / Directional / Random Search for optimal features

Embedded Method

- Part of a machine learning algorithm
- Use penalty to reduce the magnitudes of parameters