QM-7063 Data Mining
Professor: Dr. Abdulrashid
Learning Practice 8 – Noah L. Schrick

Problem 7.3 Predicting Housing Median Prices
The file BostonHousing.csv contains information on over 500 census tracts in Boston, where for each tract multiple variables are recorded. The last column (CAT.MEDV) was derived from MEDV, such that it obtains the value 1 if MEDV > 30 and 0 otherwise. Consider the goal of predicting the median value (MEDV) of a tract, given the information in the first 12 columns.

Partition the data into training (60%) and validation (40%) sets.

Data was partitioned using scikit-learn's model selection train_test_split function for a 60/40 split with random state set to 26.

 # a.
Perform a k-NN prediction with all 12 predictors (ignore the CAT.MEDV column), trying values of k from 1 to 5. Make sure to normalize the data. What is the best k? What does it mean?

Data was normalized using scikit-learn's pre-processing library. The standard scalar function was used to transform the data (after removing CAT.MEDV and MEDV since they are the outcome variables). After the data was normalized, it was concatenated together with the outcome variable to form a new, normalized data frame. From this normalized data frame, the location of the train data and validation data was identified.

For trying k values of 1 through 5, a for loop was implemented that called the KNeighborsRegressor function. Since the data contained continuous variables, the KNeighborsClassifier could not be used unless the data was first transformed.

The k-values and their accuracy are shown below:

k	Accuracy
1	64.31%
2	72.02%
3	71.06%
4	67.69%
5	66.49%

The best k is k=2. This means we have greater accuracy when each point has its nearest 2 neighbors identified.

 # b.
Predict the MEDV for a tract with the following information, using the best k:

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
0.2	0	7	0	0.538	6	62	4.7	4	307	21	10

An array with this set of data was created for the sample. Since the original data was normalized, this sample could not directly be fed into the model. This sample was first normalized using the same scalar used to normalize the original data. After normalization, this sample was put into the model.

19.45

 # c.
If we used the above k-NN algorithm to score the training data, what would be the error of the training set?

Error will be zero, or close to zero. Since the training data was used to build the model, checking the training data against itself should yield very high accuracy.

 # d.
Why is the validation data error overly optimistic compared to the error rate when applying this k-NN predictor to new data?

This model was built and tuned with a specific k value (k=2). This value of k was the best for the data used, but does not mean that this value of k is the best for future data.

 # e.
If the purpose is to predict MEDV for several thousands of new tracts, what would be the disadvantage of using k-NN prediction? List the operations that the algorithm goes through in order to produce each prediction.

1. Select k-nearest samples
2. Compute average value for k-nearest neighbors
3. Score value with new data
4. Repeat for all new data samples

If there are several thousands of new tracts, then k-NN will take much longer to run. The model will need to be rebuilt numerous times over the several thousand pieces of data. The lack of scalability will be a disadvantage.

 # Problem 8.2 Automobile Accidents
The file accidentsFull.csv contains information on 42,183 actual automobile accidents in 2001 in the United States that involved one of three levels of injury: NO INJURY, INJURY, or FATALITY. For each accident, additional information is recorded, such as day of week, weather conditions, and road type. A firm might be interested in developing a system for quickly classifying the severity of an accident based on initial reports and associated data in the system (some of which rely on GPS-assisted reporting).

Our goal here is to predict whether an accident just reported will involve an injury (MAX_SEV_IR = 1 or 2) or will not (MAX_SEV_IR = 0). For this purpose, create a dummy variable called INJURY that takes the value "yes" if MAX_SEV_IR = 1 or 2, and otherwise "no".

 # a.
Using the information in this dataset, if an accident has just been reported and no further information is available, what should the prediction be? (INJURY = Yes or No?) Why?

The data was first imported into the notebook as a dataframe to perform preliminary exploration and modification. An additional column called "Injury" was added to the dataframe, with values of '1' if the MAX_SEV_IR was greater than 0, or '0' if it was not. The mean of this column was computed.

Viewing the available data, the average value is 0.5088, where a value of 1 means all reports involved an injury, and a value of 0 means all reports did not involve an injury. With a value of 0.5088, the prediction should be 'YES' for injury, though an accident with an injury is only slightly more likely than an accident without.

 # b.
Select the first 12 records in the dataset and look only at the response (INJURY) and the two predictors WEATHER_R and TRAF_CON_R.

A subset dataframe caleld "b_records" was obtained by pulling the first 12 records of the original dataframe.

 # i.
Create a pivot table that examines INJURY as a function of the two predictors for these 12 records. Use all three variables in the pivot table as rows/columns.

WEATHER_R Pivot Table:

	WEATHER_R	1	2
Injury	x	x	x
0	x	0.333	0.667
1	x	0.667	0.333

TRAF_CON_R Pivot Table:

	TRAF_CON_R	0	1	2
Injury	x	x	x	x
0	x	0.667	0.222	0.111
1	x	1.00	0.000	0.000

 # ii.
Compute the exact Bayes conditional probabilities of an injury (INJURY = Yes) given the six possible combinations of the predictors.

For completing this question, a function was created called “computeInjuryprob”. This function took two inputs for the two predictors. This function would compute the probability of an injury given the two predictors, and would compute the probability of no injury given the two predictors. The function returned the computation of the probability of injury, divided by the sum of the injury and no injury probabilities.

P(INJURY = Yes | WEATHER_R = 1, TRAF_CON_R = 0): 0.75
P(INJURY = Yes | WEATHER_R = 2, TRAF_CON_R = 0): 0.429
P(INJURY = Yes | WEATHER_R = 1, TRAF_CON_R = 1): 0.0
P(INJURY = Yes | WEATHER_R = 2, TRAF_CON_R = 1): 0.0
P(INJURY = Yes | WEATHER_R = 1, TRAF_CON_R = 2): 0.0
P(INJURY = Yes | WEATHER_R = 2, TRAF_CON_R = 2): 0.0

 # iii.
Classify the 12 accidents using these probabilities and a cutoff of 0.5.

actual	predicted	No Injury	Injury
0	0	0.51	0.49
0	1	0.45	0.55
0	1	0.45	0.55
0	0	0.99	0.01
0	0	0.99	0.01
0	0	0.51	0.49
0	1	0.45	0.55
0	1	0.45	0.55
0	1	0.45	0.55
0	1	0.45	0.55
0	0	0.51	0.49
0	1	0.45	0.55
0	1	0.45	0.55
0	1	0.45	0.55
0	1	0.45	0.55
0	1	0.45	0.55
0	1	0.45	0.55

 # iv.
Compute manually the naive Bayes conditional probability of an injury given WEATHER_R = 1 and TRAF_CON_R = 1.

This required manually referrering to the probability and frequency tables, and computing the value.
print(2/3 * 0/3 * 3/12)

0.0, since the entire term is being multiplied by 0/3=0.

 # v.

Run a naive Bayes classifier on the 12 records and 2 predictors using scikit-learn. Check the model output to obtain probabilities and classifications for all 12 records. Compare this to the exact Bayes classification. Are the resulting classifications equivalent? Is the ranking (=ordering) of observations equivalent?

Confusion Matrix (Accuracy 0.2857)

	Prediction	
Actual	0	1
0	1	3
1	2	1

Confusion Matrix (Accuracy 0.6000)

	Prediction	
Actual	0	1
0	3	2
1	0	0

The resulting classifications are not equal, but a fair comparison cannot be drawn. Since the confusion matrix was obtained by testing the train data on the model, the model is much more likely to perform favorably.

 # c.

Let us now return to the entire dataset. Partition the data into training (60%) and validation (40%).

Data was partitioned using scikit-learn's model selection train_test_split function for a 60/40 split with random state set to 26.

 # i.

Assuming that no information or initial reports about the accident itself are available at the time of prediction (only location characteristics, weather conditions, etc.), which predictors can we include in the analysis? (Use the data descriptions page from www.dataminingbook.com.)

HOUR_I_R
ALIGN_I
WRK_ZONE
WKDY_I_R
INT_HWY
LGTCON_I_R
PROFIL_I_R
SPD_LIM
SUR_CON

TRAF_CON_R
TRAF_WAY
WEATHER_R

 # ii.
Run a naive Bayes classifier on the complete training set with the relevant predictors (and INJURY as the response). Note that all predictors are categorical. Show the confusion matrix.

The naive Bayes classifier was used with the MultinomialNB function and the predictors listed above.

Confusion Matrix (Accuracy 0.5291)

	Prediction	
Actual	0	1
0	4197	8195
1	3724	9193

 # iii.
What is the overall error for the validation set?

The same naive Bayes classifier was used to answer this question, this time scoring with the validation set rather than the training set.

Confusion Matrix (Accuracy 0.5288)

	Prediction	
Actual	0	1
0	2838	5491
1	2460	6085

 # iv.
What is the percent improvement relative to the naive rule (using the validation set)?

The percent improvement is obtained by using the accuracy from both confusion matrices as follows:
round(100* abs(0.5288 - 0.5291)/(0.5291), 3)

 # v.
Examine the conditional probabilities in the pivot tables. Why do we get a probability of zero for P(INJURY = No j SPD_LIM = 5)?

The probability is rounded to 0 due to the extremely low likelihood of sustaining an injury at such low speeds.
The pivot tables display values ranging from E-6 to E-9, which is assumed as 0.

APPENDIX

```
# Learning Practice 8 for the University of Tulsa's QM-7063 Data Mining Course
# K-Nearest Neighbor and Naive Bayes
# Professor: Dr. Abdulrashid, Spring 2023
# Noah L. Schrick – 1492657

# Imports

%matplotlib inline

from pathlib import Path

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import mean_squared_error
from math import isnan

import matplotlib.pylab as plt
from dmba import classificationSummary, gainsChart

from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn.neighbors import NearestNeighbors, KNeighborsClassifier, KneighborsRegressor

# Pre-processing
housing_df = pd.read_csv('BostonHousing.csv')
trainData, validData = train_test_split(housing_df, test_size=0.4, random_state=26)

# a
## Normalize
scaler = preprocessing.StandardScaler()

predictors = housing_df.columns.values.tolist()
predictors.remove('CAT. MEDV')
predictors.remove('MEDV')

scaler.fit(trainData[predictors])  # Note the use of an array of column names

# Transform the full dataset
housingNorm = pd.concat([pd.DataFrame(scaler.transform(housing_df[predictors]),
                                      columns=predictors),
                         housing_df[['MEDV']]], axis=1)

trainNorm = housingNorm.iloc[trainData.index]
validNorm = housingNorm.iloc[validData.index]
```

```
#newHousingNorm = pd.DataFrame(scaler.transform(housingNorm), columns=predictors)

## K-NN
results = []
train_X = trainNorm[predictors]
train_y = trainNorm['MEDV']
valid_X = validNorm[predictors]
valid_y = validNorm['MEDV']

for k in range(1,6):
    knn = KNeighborsRegressor(n_neighbors=k).fit(train_X, train_y)
    results.append({
        'k': k,
        'accuracy': knn.score(valid_X, valid_y)
    })

# Convert results to a pandas data frame
results = pd.DataFrame(results)
print(results)

# b
sample = [[0.2, 0, 7, 0, 0.538, 6, 62, 4.7, 4, 307, 21, 10]]
sample_norm = scaler.transform(sample)
ans_b = knn.predict(sample_norm)
print(ans_b)

# a.
accidents_df = pd.read_csv('accidentsFull.csv')
accidents_df['Injury'] = (accidents_df['MAX_SEV_IR'] > 0).astype(int)

accidents_df.loc[:, 'Injury'].mean()

# b.
b_records = accidents_df[0:12]

# i.
weather_freq = b_records[['Injury', 'WEATHER_R']].pivot_table(index='Injury',
columns='WEATHER_R', aggfunc=len, fill_value=0)
weather_propTable = weather_freq.apply(lambda x: x/sum(x), axis=1)

traf_freq = b_records[['Injury', 'TRAF_CON_R']].pivot_table(index='Injury',
columns='TRAF_CON_R', aggfunc=len, fill_value=0)
traf_propTable = traf_freq.apply(lambda x: x/sum(x), axis=1)

print(weather_propTable)
print(traf_propTable)
```

```
# ii.
def computeInjuryprob(wpred, tpred):
    p_w_inj = weather_propTable.iloc[1][wpred]
    p_t_inj = traf_propTable.iloc[1][tpred]
    p_inj = p_w_inj * p_t_inj

    np_w_inj = weather_propTable.iloc[0][wpred]
    np_t_inj = traf_propTable.iloc[0][tpred]
    np_inj = np_w_inj * np_t_inj

    return(p_inj/(p_inj+np_inj))

print("P(INJURY = Yes | WEATHER_R = 1, TRAF_CON_R = 0):", computeInjuryprob(1,0))
print("P(INJURY = Yes | WEATHER_R = 2, TRAF_CON_R = 0):", computeInjuryprob(2,0))
print("P(INJURY = Yes | WEATHER_R = 1, TRAF_CON_R = 1):", computeInjuryprob(1,1))
print("P(INJURY = Yes | WEATHER_R = 2, TRAF_CON_R = 1):", computeInjuryprob(2,1))
print("P(INJURY = Yes | WEATHER_R = 1, TRAF_CON_R = 2):", computeInjuryprob(1,2))
print("P(INJURY = Yes | WEATHER_R = 2, TRAF_CON_R = 2):", computeInjuryprob(2,2))

# iii.
preictors = ['WEATHER_R', 'TRAF_CON_R']
X = pd.get_dummies(b_records[predictors])
y = b_records['Injury']

# split into training and validation
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.40, random_state=1)

# run naive Bayes
delays_nb = MultinomialNB(alpha=0.01)
delays_nb.fit(X_train, y_train)

# predict probabilities
predProb_train = delays_nb.predict_proba(X_train)
predProb_valid = delays_nb.predict_proba(X_valid)

# predict class membership
y_valid_pred = delays_nb.predict(X_valid)
y_train_pred = delays_nb.predict(X_train)
# Subset a specific set
df = pd.concat([pd.DataFrame({'actual': y_valid, 'predicted': y_valid_pred}),
            pd.DataFrame(predProb_valid, index=y_valid.index)], axis=1)

for index, row in b_records.iterrows():
    mask = ((X_valid.WEATHER_R == row['WEATHER_R']) & (X_valid.TRAF_CON_R ==
row['TRAF_CON_R']))
    print(df[mask])
```

```
# iv.
print(weather_freq)
print(traf_freq)
print(2/3 * 0/3 * 3/12)

# v.
classificationSummary(y_train, y_train_pred)

print()

classificationSummary(y_valid, y_valid_pred)

# c.
trainData, validData = train_test_split(accidents_df, test_size=0.4, random_state=26)

# ii.
predictors = ['HOUR_I_R', 'ALIGN_I', 'WRK_ZONE', 'WKDY_I_R', 'INT_HWY',
              'LGTCON_I_R', 'PROFIL_I_R', 'SPD_LIM', 'SUR_COND',
              'TRAF_CON_R', 'TRAF_WAY', 'WEATHER_R']

X = pd.get_dummies(accidents_df[predictors])
y = accidents_df['Injury']

X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.40, random_state=1)

# run naive Bayes
delays_nb = MultinomialNB(alpha=0.01)
delays_nb.fit(X_train, y_train)

# predict probabilities
predProb_train = delays_nb.predict_proba(X_train)
predProb_valid = delays_nb.predict_proba(X_valid)

# predict class membership
y_valid_pred = delays_nb.predict(X_valid)
y_train_pred = delays_nb.predict(X_train)

print("Training")
classificationSummary(y_train, y_train_pred)

# iii.
print("Validation")
classificationSummary(y_valid, y_valid_pred)

# iv.
pctg_inc = round(100* abs(0.5288 - 0.5291)/(0.5291), 3)
print(pctg_inc)
```