Assignment-2

October 20, 2025

[ ]:
```
"""
Question -1 Analyze customer data from a telecom provider and build machine␣
 ↪learning models to predict churn.
Dataset: Telco Customer Churn with 7,043 customer records.
"""
```

[1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

[2]:
```
# Load dataset
df = pd.read_csv('Telco-Customer-Churn.csv')
print(df.head())
```

```
   customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1           No
1  5575-GNVDE    Male              0      No         No      34          Yes
2  3668-QPYBK    Male              0      No         No       2          Yes
3  7795-CFOCW    Male              0      No         No      45           No
4  9237-HQITU  Female              0      No         No       2          Yes

      MultipleLines InternetService OnlineSecurity  … DeviceProtection  \
0  No phone service             DSL             No  …               No
1                No             DSL            Yes  …              Yes
2                No             DSL            Yes  …               No
3  No phone service             DSL            Yes  …              Yes
4                No     Fiber optic             No  …               No

  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes
1          No          No              No        One year               No
2          No          No              No  Month-to-month              Yes
3         Yes          No              No        One year               No
4          No          No              No  Month-to-month              Yes

                PaymentMethod MonthlyCharges  TotalCharges Churn
```

```
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95        1889.5    No
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

[9]:
```
# Encode categorical variables
for col in df.select_dtypes(include='object').columns:
    if col != 'customerID':
        df[col] = LabelEncoder().fit_transform(df[col].astype(str))


print(df.head())

# Features and target
X = df.drop(columns=['customerID', 'Churn'])
y = df['Churn']
```

```
   customerID  gender  SeniorCitizen  Partner  Dependents  tenure  \
0  7590-VHVEG       0              0        1           0       1
1  5575-GNVDE       1              0        0           0      34
2  3668-QPYBK       1              0        0           0       2
3  7795-CFOCW       1              0        0           0      45
4  9237-HQITU       0              0        0           0       2

   PhoneService  MultipleLines  InternetService  OnlineSecurity  …  \
0             0              1                0               0  …
1             1              0                0               2  …
2             1              0                0               2  …
3             0              1                0               2  …
4             1              0                1               0  …

   DeviceProtection  TechSupport  StreamingTV  StreamingMovies  Contract  \
0                 0            0            0                0         0
1                 2            0            0                0         1
2                 0            0            0                0         0
3                 2            2            0                0         1
4                 0            0            0                0         0

   PaperlessBilling  PaymentMethod  MonthlyCharges  TotalCharges  Churn
0                 1              2           29.85          2505      0
1                 0              3           56.95          1466      0
2                 1              3           53.85           157      1
3                 0              0           42.30          1400      0
4                 1              2           70.70           925      1
```

```
[5 rows x 21 columns]
```

[7]:
```python
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Train model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.7984386089425124
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1036
           1       0.66      0.49      0.56       373

    accuracy                           0.80      1409
   macro avg       0.75      0.70      0.71      1409
weighted avg       0.79      0.80      0.79      1409
```

[11]:
```python
# Example customer data (must have same features as X_train)
sample_data = pd.DataFrame([{
    'gender': 1,                  # 1 = Male, 0 = Female (encoded)
    'SeniorCitizen': 0,
    'Partner': 1,
    'Dependents': 0,
    'tenure': 5,
    'PhoneService': 1,
    'MultipleLines': 0,
    'InternetService': 1,         # 0=DSL, 1=Fiber optic, 2=No (encoded)
    'OnlineSecurity': 0,
    'OnlineBackup': 0,
    'DeviceProtection': 0,
    'TechSupport': 0,
    'StreamingTV': 1,
    'StreamingMovies': 1,
    'Contract': 0,                # 0=Month-to-month, 1=One year, 2=Two year
    'PaperlessBilling': 1,
    'PaymentMethod': 3,           # depends on encoding, e.g. 3=Electronic check
    'MonthlyCharges': 85.5,
    'TotalCharges': 400.5
}])
```

```
# Predict churn for the sample
prediction = model.predict(sample_data)

if prediction[0] == 1:
    print(" Customer is likely to CHURN.")
else:
    print(" Customer is NOT likely to churn.")
```

```
 Customer is likely to CHURN.
```

[ ]:
```
""" Question -2 : Apply parallel tree boosting algorithms (such as XGBoost,␣
 ↪LightGBM, or CatBoost) to a real-world dataset to predict customer
default behavior.
Dataset: UCI Credit Card Default Dataset with 30,000 records. """
```

[13]:
```
import lightgbm as lgb
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

[15]:
```
df = pd.read_csv("UCI_Credit_Card.csv")

print("Dataset shape:", df.shape)
print(df.head())

# Target variable
target = 'default.payment.next.month'

# Features & Target
X = df.drop(columns=[target])
y = df[target]

# Split into train/test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
Dataset shape: (30000, 25)
   ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_0  PAY_2  PAY_3  PAY_4  \
0   1    20000.0    2          2         1   24      2      2     -1     -1
1   2   120000.0    2          2         2   26     -1      2      0      0
2   3    90000.0    2          2         2   34      0      0      0      0
3   4    50000.0    2          2         1   37      0      0      0      0
```

```
4   5    50000.0    1          2         1   57     -1      0     -1      0

   …  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  \
0  …        0.0        0.0        0.0       0.0     689.0       0.0
1  …     3272.0     3455.0     3261.0       0.0    1000.0    1000.0
2  …    14331.0    14948.0    15549.0    1518.0    1500.0    1000.0
3  …    28314.0    28959.0    29547.0    2000.0    2019.0    1200.0
4  …    20940.0    19146.0    19131.0    2000.0   36681.0   10000.0

   PAY_AMT4  PAY_AMT5  PAY_AMT6  default.payment.next.month
0       0.0       0.0       0.0                          1
1    1000.0       0.0    2000.0                          1
2    1000.0    1000.0    5000.0                          0
3    1100.0    1069.0    1000.0                          0
4    9000.0     689.0     679.0                          0

[5 rows x 25 columns]
```

[19]:
```
xgb_model = XGBClassifier(
    n_estimators=200,
    learning_rate=0.1,
    max_depth=5,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    eval_metric='logloss'
)
xgb_model.fit(X_train, y_train)
pred_xgb = xgb_model.predict(X_test)
acc_xgb = accuracy_score(y_test, pred_xgb)
```

[21]:
```
lgb_model = lgb.LGBMClassifier(
    n_estimators=200,
    learning_rate=0.1,
    num_leaves=31,
    random_state=42
)
lgb_model.fit(X_train, y_train)
pred_lgb = lgb_model.predict(X_test)
acc_lgb = accuracy_score(y_test, pred_lgb)
```

```
[LightGBM] [Info] Number of positive: 5323, number of negative: 18677
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.001423 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 3519
[LightGBM] [Info] Number of data points in the train set: 24000, number of used
```

```
features: 24
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.221792 -> initscore=-1.255256
[LightGBM] [Info] Start training from score -1.255256
```

[23]:
```
cat_model = CatBoostClassifier(
    iterations=200,
    learning_rate=0.1,
    depth=6,
    verbose=0,
    random_state=42
)
cat_model.fit(X_train, y_train)
pred_cat = cat_model.predict(X_test)
acc_cat = accuracy_score(y_test, pred_cat)
```

[29]:
```
print("\nMODEL PERFORMANCE COMPARISON")
print(f"XGBoost Accuracy  : {acc_xgb:.4f}")
print(f"LightGBM Accuracy  : {acc_lgb:.4f}")
print(f"CatBoost Accuracy  : {acc_cat:.4f}")
```

```
MODEL PERFORMANCE COMPARISON
XGBoost Accuracy  : 0.8233
LightGBM Accuracy  : 0.8197
CatBoost Accuracy  : 0.8197
```

[ ]:
```
"""
Question : 3 Predict traffic congestion levels public APIs (e.g., Google␣
 ↪Traffic, OpenTraffic) data or real-time traffic data from city
traffic sensors or open datasets.
"""
```

[37]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
import xgboost as xgb
```

[39]:
```
df = pd.read_csv("Traffic.csv")
print(df.head())
```

```
          Time  Date Day of the week  CarCount  BikeCount  BusCount  \
0  12:00:00 AM    10          Tuesday        31          0         4
1  12:15:00 AM    10          Tuesday        49          0         3
2  12:30:00 AM    10          Tuesday        46          0         3
3  12:45:00 AM    10          Tuesday        51          0         2
4   1:00:00 AM    10          Tuesday        57          6        15

   TruckCount  Total Traffic Situation
```

```
0           4     39              low
1           3     55              low
2           6     55              low
3           5     58              low
4          16     94           normal
```

[47]:
```
# --- Preprocessing ---
# Convert 'Time' to numeric hour
df['Time'] = pd.to_datetime(df['Time']).dt.hour

# Encode 'Day of the week'
le_day = LabelEncoder()
df['Day of the week'] = le_day.fit_transform(df['Day of the week'])

# Encode target 'Situation'
le_situation = LabelEncoder()
df['Traffic Situation'] = le_situation.fit_transform(df['Traffic Situation'])

# --- Feature & Target ---
X = df[['Time','Date','Day of the␣
 ↪week','CarCount','BikeCount','BusCount','TruckCount','Total']]
y = df['Traffic Situation']
```

[59]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)


model = xgb.XGBClassifier( n_estimators=200,learning_rate=0.
 ↪1,max_depth=15,eval_metric='mlogloss')
model.fit(X_train, y_train)


y_pred = model.predict(X_test)
print("\nModel Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")

class_names = [str(cls) for cls in le_situation.
 ↪inverse_transform(range(len(le_situation.classes_)))]
print(classification_report(y_test, y_pred, target_names=class_names))
```

```
Model Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       197
           1       1.00      1.00      1.00        88
```

```
           2       1.00      1.00      1.00        93
           3       1.00      1.00      1.00       515

    accuracy                           1.00       893
   macro avg       1.00      1.00      1.00       893
weighted avg       1.00      1.00      1.00       893
```

[ ]:
```
"""
Question : 4
Forecast electricity usage in smart homes.
Dataset: UCI Individual Household Electric Power Consumption dataset.
"""
```

[77]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor, StackingRegressor
from xgboost import XGBRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

[79]:
```
data = pd.read_csv('household_power_consumption.txt',
                   sep=';',
                   parse_dates={'datetime': ['Date', 'Time']},
                   infer_datetime_format=True,
                   na_values='?',
                   low_memory=False)

data.fillna(method='ffill', inplace=True)

data.set_index('datetime', inplace=True)

cols =␣
 ↪['Global_active_power','Global_reactive_power','Voltage','Global_intensity',
       'Sub_metering_1','Sub_metering_2','Sub_metering_3']
data[cols] = data[cols].apply(pd.to_numeric)

data_hourly = data.resample('H').mean()

data_hourly['hour'] = data_hourly.index.hour
data_hourly['day_of_week'] = data_hourly.index.dayofweek
data_hourly['month'] = data_hourly.index.month

data_hourly['lag_1'] = data_hourly['Global_active_power'].shift(1)
data_hourly['lag_24'] = data_hourly['Global_active_power'].shift(24)
```

```
data_hourly.dropna(inplace=True)
```

```
C:\Users\Uttkarsh Dhiman\AppData\Local\Temp\ipykernel_18056\3084884950.py:1:
FutureWarning: Support for nested sequences for 'parse_dates' in pd.read_csv is
deprecated. Combine the desired columns with pd.to_datetime after parsing
instead.
  data = pd.read_csv('household_power_consumption.txt',
C:\Users\Uttkarsh Dhiman\AppData\Local\Temp\ipykernel_18056\3084884950.py:1:
FutureWarning: The argument 'infer_datetime_format' is deprecated and will be
removed in a future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.
  data = pd.read_csv('household_power_consumption.txt',
C:\Users\Uttkarsh Dhiman\AppData\Local\Temp\ipykernel_18056\3084884950.py:1:
UserWarning: Parsing dates in %d/%m/%Y %H:%M:%S format when dayfirst=False (the
default) was specified. Pass `dayfirst=True` or specify a format to silence this
warning.
  data = pd.read_csv('household_power_consumption.txt',
C:\Users\Uttkarsh Dhiman\AppData\Local\Temp\ipykernel_18056\3084884950.py:8:
FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a
future version. Use obj.ffill() or obj.bfill() instead.
  data.fillna(method='ffill', inplace=True)
C:\Users\Uttkarsh Dhiman\AppData\Local\Temp\ipykernel_18056\3084884950.py:16:
FutureWarning: 'H' is deprecated and will be removed in a future version, please
use 'h' instead.
  data_hourly = data.resample('H').mean()
```

[81]:
```
X = data_hourly.drop('Global_active_power', axis=1)
y = data_hourly['Global_active_power']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪shuffle=False)
```

[93]:
```
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)

rmse = mean_squared_error(y_test, y_pred)
print(f"Random Forest RMSE: {rmse:.3f}")
```

```
Random Forest RMSE: 0.000
```

[94]:
```
sample_input = pd.DataFrame({
    'Global_reactive_power': [0.1],    # previous hour's reactive power
    'Voltage': [235.0],                 # voltage
    'Global_intensity': [5.0],          # current intensity
    'Sub_metering_1': [0.0],            # kitchen
```

```
    'Sub_metering_2': [1.0],           # laundry
    'Sub_metering_3': [2.0],           # water heater/AC
    'hour': [14],                       # 2 PM
    'day_of_week': [2],                 # Wednesday
    'month': [10],                       # October
    'lag_1': [1.2],                     # Global_active_power 1 hour ago
    'lag_24': [1.0]                     # Global_active_power 24 hours ago
})

# Predict using trained Random Forest
predicted_power = rf.predict(sample_input)
print(f"Predicted Global Active Power: {predicted_power[0]:.3f} kW")
```

```
Predicted Global Active Power: 1.159 kW
```

[ ]:
```
"""
Question 5) Classify product or service reviews as positive/negative/neutral.
Dataset used : Dafiniti_hotel _reviews.
"""
```

[101]:
```
hotel_review = pd.read_csv("Datafiniti_Hotel_Reviews.csv")
print(hotel_review.head())
```

```
                    id             dateAdded           dateUpdated  \
0  AVwc252WIN2L1WUfpqLP  2016-10-30T21:42:42Z  2018-09-10T21:06:27Z
1  AVwc252WIN2L1WUfpqLP  2016-10-30T21:42:42Z  2018-09-10T21:06:27Z
2  AVwc252WIN2L1WUfpqLP  2016-10-30T21:42:42Z  2018-09-10T21:06:27Z
3  AVwdOclqIN2L1WUfti38  2015-11-28T19:19:35Z  2018-09-10T21:06:16Z
4  AVwdOclqIN2L1WUfti38  2015-11-28T19:19:35Z  2018-09-10T21:06:16Z

            address                                          categories  \
0  5921 Valencia Cir  Hotels,Hotels and motels,Hotel and motel reser…
1  5921 Valencia Cir  Hotels,Hotels and motels,Hotel and motel reser…
2  5921 Valencia Cir  Hotels,Hotels and motels,Hotel and motel reser…
3     7520 Teague Rd  Hotels,Hotels and motels,Travel agencies and b…
4     7520 Teague Rd  Hotels,Hotels and motels,Travel agencies and b…

              primaryCategories             city country  \
0  Accommodation & Food Services  Rancho Santa Fe      US
1  Accommodation & Food Services  Rancho Santa Fe      US
2  Accommodation & Food Services  Rancho Santa Fe      US
3  Accommodation & Food Services          Hanover      US
4  Accommodation & Food Services          Hanover      US

                                          keys   latitude  …  \
0  us/ca/ranchosantafe/5921valenciacir/359754519  32.990959  …
1  us/ca/ranchosantafe/5921valenciacir/359754519  32.990959  …
2  us/ca/ranchosantafe/5921valenciacir/359754519  32.990959  …
```

```
3         us/md/hanover/7520teaguerd/-2043779672  39.155929  …
4         us/md/hanover/7520teaguerd/-2043779672  39.155929  …

                                  reviews.dateSeen reviews.rating  \
0  2016-08-03T00:00:00Z,2016-07-26T00:00:00Z,2016…            5.0
1  2016-08-02T00:00:00Z,2016-08-26T00:00:00Z,2016…            5.0
2  2016-11-15T00:00:00Z,2016-08-23T00:00:00Z,2016…            5.0
3          2016-05-21T00:00:00Z,2016-07-31T00:00:00Z            2.0
4                              2016-07-31T00:00:00Z            5.0

                                reviews.sourceURLs  \
0   https://www.hotels.com/hotel/125419/reviews%20/
1   https://www.hotels.com/hotel/125419/reviews%20/
2   https://www.hotels.com/hotel/125419/reviews%20/
3  https://www.tripadvisor.com/Hotel_Review-g4118…
4  https://www.tripadvisor.com/Hotel_Review-g4118…

                                     reviews.text  \
0  Our experience at Rancho Valencia was absolute…
1  Amazing place. Everyone was extremely warm and…
2  We booked a 3 night stay at Rancho Valencia to…
3  Currently in bed writing this for the past hr …
4  I live in Md and the Aloft is my Home away fro…

                             reviews.title reviews.userCity  \
0          Best romantic vacation ever!!!!              NaN
1                    Sweet sweet serenity              NaN
2          Amazing Property and Experience              NaN
3  Never again…beware, if you want sleep.         Richmond
4                    ALWAYS GREAT STAY…           Laurel

   reviews.userProvince reviews.username  \
0                   NaN            Paula
1                   NaN                D
2                   NaN              Ron
3                    VA        jaeem2016
4                    MD       MamaNiaOne

                                        sourceURLs  \
0  http://www.hotels.com/ho125419/%25252525253Flo…
1  http://www.hotels.com/ho125419/%25252525253Flo…
2  http://www.hotels.com/ho125419/%25252525253Flo…
3  http://www.yellowbook.com/profile/aloft-arunde…
4  http://www.yellowbook.com/profile/aloft-arunde…

                                     websites
0                    http://www.ranchovalencia.com
1                    http://www.ranchovalencia.com
```

```
2                        http://www.ranchovalencia.com
3  http://www.starwoodhotels.com/alofthotels/prop…
4  http://www.starwoodhotels.com/alofthotels/prop…

[5 rows x 25 columns]
```

[103]:
```python
def map_sentiment(rating):
    if rating <= 2:
        return 'negative'
    elif rating == 3:
        return 'neutral'
    else:
        return 'positive'

hotel_review['sentiment'] = hotel_review['reviews.rating'].apply(map_sentiment)

# Keep only review text and sentiment
data = hotel_review[['reviews.text', 'sentiment']].dropna()
```

[111]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer
X = data['reviews.text']
y = data['sentiment']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# ----------------------------
# 5  Text Vectorization
# ----------------------------
vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

[115]:
```python
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train_vec, y_train)
y_pred = clf.predict(X_test_vec)
```

[119]:
```python
from sklearn.metrics import classification_report, confusion_matrix
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))

sample_review = ["The hotel was amazing and the staff was very friendly"]
sample_vec = vectorizer.transform(sample_review)
sample_pred = clf.predict(sample_vec)
print(f"\nSample Review Prediction: {sample_pred[0]}")
```

```
Classification Report:

              precision    recall  f1-score   support

    negative       0.75      0.37      0.50       231
     neutral       0.56      0.02      0.04       237
    positive       0.81      0.99      0.89      1532

    accuracy                           0.80      2000
   macro avg       0.70      0.46      0.48      2000
weighted avg       0.77      0.80      0.74      2000


Sample Review Prediction: positive
```

[ ]: