make_final_prediction_csv

May 2, 2024

CIS 662: INTRO TO MACHINE LEARNING AND ALGORITHMS

Semester Project

Group 13: - Niranjan Balasubramani - Preet Karia - Spoorthi Jayaprakash Malgund - Parth Pramod Kulkarni

[1]:
```
# Generic inputs for most ML tasks
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn import tree
import graphviz
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
import xgboost as xgb

pd.options.display.float_format = '{:,.2f}'.format
pd.set_option('display.max_columns', None)

# setup interactive notebook mode
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

from IPython.display import display, HTML
```

```
/var/folders/tl/lnf2sv191t77b2f4hhxqlcx80000gn/T/ipykernel_98551/3709173474.py:2
: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of
pandas (pandas 3.0),
```

```
(to allow more performant data types, such as the Arrow string type, and better
interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

0.1 Load all datasets

[2]:
```
submission_csv = pd.read_csv('../test_data/Professor_CSV.csv',␣
 ↪keep_default_na=False)
former_flights_data = pd.read_csv('../dataset/merged_data/former_flight_data.
 ↪csv')
latter_flight_data = pd.read_csv('../dataset/merged_data/latter_flight_data.
 ↪csv')
test_data = pd.read_csv('../test_data/Test_Merged_Data.csv')
```

0.2 Preprocess all datasets

[3]:
```
if True:
    latter_flight_data.dropna(subset=['FORMER_FLIGHT_STATUS'], inplace=True)
```

[4]:
```
def delay_categories(delay):
    if delay < -5:
        # early
        return 0
    elif delay > 5:
        # late
        return 2
    else:
        # on-time
        return 1
```

[5]:
```
X_former_flights_data = former_flights_data.drop(columns=['ARR_DELAY'])
y_former_flights_data = former_flights_data['ARR_DELAY'].apply(delay_categories)

X_latter_flight_data = latter_flight_data.drop(columns=['ARR_DELAY'])
y_latter_flight_data = latter_flight_data['ARR_DELAY'].apply(delay_categories)
```

[6]:
```
categorical_vars = ['DAY_OF_WEEK', 'MKT_UNIQUE_CARRIER',
                    'OP_UNIQUE_CARRIER', 'ORIGIN',
                    'ORGIN_WTH_precipprob', 'ORGIN_WTH_severerisk',
                    'DEST_WTH_precipprob', 'DEST_WTH_severerisk',
                    'FORMER_FLIGHT_STATUS', 'MONTH']
```

[7]:
```
def preprocess(flight_data: pd.DataFrame):

    # Dealing with date and time
    flight_data['SCH_ARR_TIME'] = pd.to_datetime(flight_data['SCH_ARR_TIME'])
    flight_data['SCH_DEP_TIME'] = pd.to_datetime(flight_data['SCH_DEP_TIME'])

    flight_data['MONTH'] = flight_data['SCH_ARR_TIME'].dt.month
    flight_data['DAY'] = flight_data['SCH_ARR_TIME'].dt.day
    flight_data['DEP_MINUTES'] = flight_data['SCH_DEP_TIME'].dt.hour * 60 +␣
 ↪flight_data['SCH_DEP_TIME'].dt.minute
    flight_data['ARR_MINUTES'] = flight_data['SCH_ARR_TIME'].dt.hour * 60 +␣
 ↪flight_data['SCH_ARR_TIME'].dt.minute

    flight_data.drop(columns=['SCH_DEP_TIME', 'SCH_ARR_TIME'], inplace=True)

    # Dropping unwanted columns
    cols = [
        'ORGIN_WTH_temp', 'DEST_WTH_temp',
        'DEST_WTH_severerisk', 'ORGIN_WTH_severerisk',
        'DEST_WTH_precipprob', 'ORGIN_WTH_precipprob'
        ]
    flight_data.drop(columns=cols, inplace=True)

    cat_col = list(set(flight_data.columns).intersection(categorical_vars))
    flight_data = pd.get_dummies(flight_data, columns = list(cat_col),␣
 ↪drop_first = False)

    return flight_data
```

[8]:
```
X_former_flights_data = preprocess(X_former_flights_data)
X_latter_flight_data = preprocess(X_latter_flight_data)
test_data = preprocess(test_data)
```

[9]:
```
# Trying SMOTE
if False:
    import imblearn
    from imblearn.over_sampling import SMOTE
    from imblearn.pipeline import Pipeline
    from imblearn.under_sampling import RandomUnderSampler
    from collections import Counter

    over = SMOTE()
    under = RandomUnderSampler()
    steps = [('o', over), ('u', under)]
    pipeline = Pipeline(steps=steps)
```

```
    X_former_flights_data, y_former_flights_data = pipeline.
 ↪fit_resample(X_former_flights_data, y_former_flights_data)
    X_latter_flight_data, y_latter_flight_data = pipeline.
 ↪fit_resample(X_latter_flight_data, y_latter_flight_data)
```

[10]:
```
# Get missing columns in the prediction data
missing_cols = set(X_former_flights_data.columns) - set(test_data.columns)
# Add a zero column for missing columns in prediction data
for c in missing_cols:
    test_data[c] = 0

# Ensure the order of columns in prediction data matches that of␣
 ↪flight_data_encoded
test_data = test_data[X_former_flights_data.columns]
test_data.columns
test_data.shape
# Now, prediction_data_encoded should have the same columns as␣
 ↪flight_data_encoded
```

[10]:
```
Index(['ORGIN_WTH_precip', 'ORGIN_WTH_snow', 'ORGIN_WTH_windspeed',
       'ORGIN_WTH_winddir', 'ORGIN_WTH_cloudcover', 'ORGIN_WTH_visibility',
       'DEST_WTH_precip', 'DEST_WTH_snow', 'DEST_WTH_windspeed',
       'DEST_WTH_winddir', 'DEST_WTH_cloudcover', 'DEST_WTH_visibility', 'DAY',
       'DEP_MINUTES', 'ARR_MINUTES', 'MKT_UNIQUE_CARRIER_AA',
       'MKT_UNIQUE_CARRIER_B6', 'MKT_UNIQUE_CARRIER_DL',
       'MKT_UNIQUE_CARRIER_UA', 'MKT_UNIQUE_CARRIER_WN', 'MONTH_1', 'MONTH_2',
       'MONTH_3', 'MONTH_4', 'MONTH_5', 'MONTH_6', 'MONTH_7', 'MONTH_8',
       'MONTH_9', 'MONTH_10', 'MONTH_11', 'MONTH_12', 'OP_UNIQUE_CARRIER_9E',
       'OP_UNIQUE_CARRIER_B6', 'OP_UNIQUE_CARRIER_G7', 'OP_UNIQUE_CARRIER_MQ',
       'OP_UNIQUE_CARRIER_OO', 'OP_UNIQUE_CARRIER_PT', 'OP_UNIQUE_CARRIER_UA',
       'OP_UNIQUE_CARRIER_WN', 'OP_UNIQUE_CARRIER_YX', 'OP_UNIQUE_CARRIER_ZW',
       'ORIGIN_JFK', 'ORIGIN_MCO', 'ORIGIN_ORD', 'DAY_OF_WEEK_1',
       'DAY_OF_WEEK_2', 'DAY_OF_WEEK_3', 'DAY_OF_WEEK_4', 'DAY_OF_WEEK_5',
       'DAY_OF_WEEK_6', 'DAY_OF_WEEK_7'],
      dtype='object')
```

[10]:
```
(23, 52)
```

0.3 Train Former Flight and Latter Flight Model

[11]:
```
former_flight_model = xgb.XGBClassifier(eta = '0.006', max_depth=4,␣
 ↪min_child_weight=2, n_estimators=600, reg_alpha=0.006, reg_lambda=0.009)

former_flight_model = former_flight_model.fit(X_former_flights_data,␣
 ↪y_former_flights_data)
former_flight_model.score(X_former_flights_data, y_former_flights_data)
```

```
feat_imp_former = pd.Series(former_flight_model.feature_importances_,␣
 ↪X_former_flights_data.columns.values).sort_values(ascending=False)
feat_imp_former.head(15)



latter_flight_model = xgb.XGBClassifier(eta = '0.006', max_depth=4,␣
 ↪min_child_weight=2, n_estimators=600, reg_alpha=0.006, reg_lambda=0.009)

latter_flight_model = latter_flight_model.fit(X_latter_flight_data,␣
 ↪y_latter_flight_data)
latter_flight_model.score(X_latter_flight_data, y_latter_flight_data)
feat_imp_latter = pd.Series(latter_flight_model.feature_importances_,␣
 ↪X_latter_flight_data.columns.values).sort_values(ascending=False)
feat_imp_latter.head(15)
```

[11]: 0.5651852945518973

[11]:
```
MKT_UNIQUE_CARRIER_B6   0.11
MKT_UNIQUE_CARRIER_DL   0.10
ORIGIN_ORD              0.05
ORGIN_WTH_visibility    0.03
ORGIN_WTH_snow          0.03
OP_UNIQUE_CARRIER_YX    0.03
ORGIN_WTH_precip        0.03
DAY_OF_WEEK_5           0.03
OP_UNIQUE_CARRIER_UA    0.03
OP_UNIQUE_CARRIER_MQ    0.02
ORGIN_WTH_cloudcover    0.02
ARR_MINUTES             0.02
DEP_MINUTES             0.02
MONTH_1                 0.02
DEST_WTH_snow           0.02
dtype: float32
```

[11]: 0.5788250211327134

[11]:
```
MKT_UNIQUE_CARRIER_B6        0.07
MKT_UNIQUE_CARRIER_DL        0.06
OP_UNIQUE_CARRIER_UA         0.04
ORGIN_WTH_snow               0.04
ORGIN_WTH_visibility         0.03
ARR_MINUTES                  0.03
FORMER_FLIGHT_STATUS_early   0.03
DAY_OF_WEEK_5                0.03
MONTH_1                      0.02
DEP_MINUTES                  0.02
```

```
DAY_OF_WEEK_3                0.02
ORGIN_WTH_precip             0.02
ORGIN_WTH_cloudcover         0.02
OP_UNIQUE_CARRIER_OO         0.02
OP_UNIQUE_CARRIER_ZW         0.02
dtype: float32
```

0.4 Make Predictions and write to csv

[12]:
```
status_dic = {0: 'early', 1: 'ontime', 2: 'late'}
for index, sub_row in submission_csv.iterrows():
    test_row = test_data.iloc[index].copy()

    # Predict Former
    former = status_dic[former_flight_model.predict([test_row])[0]]
    if sub_row['ARRIVAL STATUS'] != 'NA':
        sub_row['ARRIVAL STATUS'] = former

    # Predict Latter - Former Early
    test_row['FORMER_FLIGHT_STATUS_late'] = 0
    test_row['FORMER_FLIGHT_STATUS_on-time'] = 0
    test_row['FORMER_FLIGHT_STATUS_early'] = 1
    early = status_dic[latter_flight_model.predict([test_row])[0]]
    if sub_row['ARRIVAL STATUS_Prev_flight_early'] != 'NA':
        sub_row['ARRIVAL STATUS_Prev_flight_early'] = early

    # Predict Latter - Former ontime
    test_row['FORMER_FLIGHT_STATUS_late'] = 0
    test_row['FORMER_FLIGHT_STATUS_on-time'] = 1
    test_row['FORMER_FLIGHT_STATUS_early'] = 0
    ontime = status_dic[latter_flight_model.predict([test_row])[0]]
    if sub_row['ARRIVAL STATUS_Prev_flight_ontime'] != 'NA':
        sub_row['ARRIVAL STATUS_Prev_flight_ontime'] = ontime

    # Predict Latter - Former late
    test_row['FORMER_FLIGHT_STATUS_late'] = 1
    test_row['FORMER_FLIGHT_STATUS_on-time'] = 0
    test_row['FORMER_FLIGHT_STATUS_early'] = 0
    late = status_dic[latter_flight_model.predict([test_row])[0]]
    if sub_row['ARRIVAL STATUS_Prev_flight_late'] != 'NA':
        sub_row['ARRIVAL STATUS_Prev_flight_late'] = late

submission_csv.head()
```

[12]:
```
       DATE     DAY FLIGHT NUMBER MKT_UNIQUE_CARRIER OP_UNIQUE_CARRIER ORIGIN  \
0  4/19/24  FRIDAY       UA 1400                  UA                 UA    ORD
```

```
1  4/19/24  FRIDAY       AA 3402                AA                MQ    ORD
2  4/19/24  FRIDAY        B6 116                B6                B6    JFK
3  4/19/24  FRIDAY       DL 5182                DL                9E    JFK
4  4/19/24  FRIDAY       WN 5285                WN                WN    MCO

  DEPARTURE TIME ARRIVAL TIME ARRIVAL STATUS ARRIVAL STATUS_Prev_flight_early  \
0        6:52 PM      9:47 PM          early                                NA
1        7:59 PM     10:52 PM             NA                             early
2        1:34 PM      2:51 PM           late                                NA
3        2:55 PM      4:21 PM             NA                             early
4       11:35 AM      2:20 PM           late                                NA

  ARRIVAL STATUS_Prev_flight_ontime ARRIVAL STATUS_Prev_flight_late
0                                NA                             NA
1                             early                          early
2                                NA                             NA
3                             early                           late
4                                NA                             NA
```

[13]:
```
submission_csv.drop(columns=['MKT_UNIQUE_CARRIER', 'OP_UNIQUE_CARRIER'],␣
↪inplace=True)
```

[14]:
```
submission_csv.to_csv('./Group_13_Submission_CSV.csv', index=False)
```

[15]:
```
import datetime

def get_status(scheduled, actual):
    time_format = '%I:%M %p'  # Format for hours:minutes AM/PM
    scheduled_time = datetime.datetime.strptime(scheduled, time_format)
    actual_time = datetime.datetime.strptime(actual, time_format)

    delay = actual_time - scheduled_time

    if delay < datetime.timedelta(minutes=-5):
        return 'early'
    elif delay > datetime.timedelta(minutes=5):
        return 'late'
    else:
        return 'ontime'
```

[16]:
```
actual_df = pd.read_csv('../test_data/Actual_Test_Data.csv')
actual_df = actual_df.head(submission_csv.shape[0])
if "ACTUAL ARRIVAL TIME" in actual_df.columns:
    check_correctness_df = submission_csv.merge(actual_df['ACTUAL ARRIVAL␣
↪TIME'], left_index=True, right_index=True)
    check_correctness_df['Actual Status'] = check_correctness_df['ARRIVAL TIME']
```

```
    check_correctness_df['Actual Status'] = check_correctness_df.apply(lambda␣
 ↪row: get_status(row['ARRIVAL TIME'], row['ACTUAL ARRIVAL TIME']), axis=1)
else:
    check_correctness_df = submission_csv.merge(actual_df['Actual Status'],␣
 ↪left_index=True, right_index=True)



count = 0
for i in range(0, check_correctness_df.shape[0]):
    cur_row = check_correctness_df.iloc[i]

    # former
    if cur_row['ARRIVAL STATUS'] != 'NA':
        if cur_row['ARRIVAL STATUS'] != cur_row['Actual Status']: count += 1
        former_status = cur_row['Actual Status']

    # latter
    else:
        col = 'ARRIVAL STATUS_Prev_flight_' + former_status
        if cur_row[col] != cur_row['Actual Status']: count += 1

print(f'Correct: {check_correctness_df.shape[0] - count}\nIncorrect: {count}')

check_correctness_df
```

```
Correct: 9
Incorrect: 14
```

[16]:

	DATE	DAY	FLIGHT NUMBER	ORIGIN	DEPARTURE TIME	ARRIVAL TIME	\
0	4/19/24	FRIDAY	UA 1400	ORD	6:52 PM	9:47 PM	
1	4/19/24	FRIDAY	AA 3402	ORD	7:59 PM	10:52 PM	
2	4/19/24	FRIDAY	B6 116	JFK	1:34 PM	2:51 PM	
3	4/19/24	FRIDAY	DL 5182	JFK	2:55 PM	4:21 PM	
4	4/19/24	FRIDAY	WN 5285	MCO	11:35 AM	2:20 PM	
5	4/19/24	FRIDAY	B6 656	MCO	1:35 PM	4:25 PM	
6	4/20/24	SATURDAY	UA 1400	ORD	6:52 PM	9:47 PM	
7	4/20/24	SATURDAY	AA 3402	ORD	7:59 PM	10:52 PM	
8	4/20/24	SATURDAY	B6 116	JFK	1:25 PM	2:41 PM	
9	4/20/24	SATURDAY	DL 5182	JFK	2:55 PM	4:21 PM	
10	4/20/24	SATURDAY	B6 656	MCO	1:35 PM	4:25 PM	
11	4/21/24	SUNDAY	UA 1400	ORD	6:52 PM	9:47 PM	
12	4/21/24	SUNDAY	AA 3402	ORD	7:59 PM	10:52 PM	
13	4/21/24	SUNDAY	B6 116	JFK	1:35 PM	2:51 PM	
14	4/21/24	SUNDAY	DL 5182	JFK	2:55 PM	4:21 PM	
15	4/21/24	SUNDAY	WN 5285	MCO	11:05 AM	1:50 PM	
16	4/21/24	SUNDAY	B6 656	MCO	1:35 PM	4:25 PM	

17	4/22/24	MONDAY	UA 1400	ORD	6:52 PM	9:47 PM
18	4/22/24	MONDAY	AA 3402	ORD	7:59 PM	10:52 PM
19	4/22/24	MONDAY	B6 116	JFK	1:35 PM	2:51 PM
20	4/22/24	MONDAY	DL 5182	JFK	2:55 PM	4:21 PM
21	4/22/24	MONDAY	WN 5285	MCO	11:35 AM	2:20 PM
22	4/22/24	MONDAY	B6 656	MCO	1:34 PM	4:25 PM

	ARRIVAL STATUS	ARRIVAL STATUS_Prev_flight_early
0	early	NA
1	NA	early
2	late	NA
3	NA	early
4	late	NA
5	NA	early
6	early	NA
7	NA	late
8	late	NA
9	NA	early
10	late	NA
11	early	NA
12	NA	late
13	late	NA
14	NA	early
15	late	NA
16	NA	early
17	late	NA
18	NA	late
19	early	NA
20	NA	early
21	late	NA
22	NA	early

	ARRIVAL STATUS_Prev_flight_ontime	ARRIVAL STATUS_Prev_flight_late
0	NA	NA
1	early	early
2	NA	NA
3	early	late
4	NA	NA
5	early	early
6	NA	NA
7	late	late
8	NA	NA
9	early	early
10	NA	NA
11	NA	NA
12	late	late
13	NA	NA

14	early	early
15	NA	NA
16	early	early
17	NA	NA
18	late	late
19	NA	NA
20	early	early
21	NA	NA
22	early	early

	Actual Status
0	late
1	late
2	early
3	early
4	ontime
5	early
6	early
7	early
8	early
9	ontime
10	early
11	early
12	late
13	late
14	early
15	early
16	late
17	early
18	ontime
19	early
20	early
21	early
22	ontime

[ ]: