In [114…
```
import pandas as pd
import matplotlib.pylab as plt
import numpy as np
import seaborn as sns
from numpy import exp

import statistics as stats

from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import pairwise
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression, LogisticRegression, Logis
from sklearn.model_selection import train_test_split, cross_val_score, GridS
from sklearn.metrics import f1_score, precision_score, recall_score, accurac

from imblearn.over_sampling import RandomOverSampler

import dmba
from dmba import regressionSummary, plotDecisionTree, classificationSummary,

%matplotlib inline
```

In [115…
```
absent = pd.read_csv('/Users/mtc/ADS/ADS 505/Project/Absenteeism_at_work.csv
# absent = pd.read_csv('/Users/patriciomartinez/Downloads/absenteeism+at+wor
```

In [116…
```
absent.head()
```

Out[116…

	ID	Reason for absence	Month of absence	Day of the week	Seasons	Transportation expense	Distance from Residence to Work	Service time	Age
0	11	26	7	3	1	289	36	13	33
1	36	0	7	3	1	118	13	18	50
2	3	23	7	4	1	179	51	18	38
3	7	7	7	5	1	279	5	14	39
4	11	23	7	5	1	289	36	13	33

5 rows × 21 columns

In [117…
```
absent.columns
```

Out[117…
```
Index(['ID', 'Reason for absence', 'Month of absence', 'Day of the week',
       'Seasons', 'Transportation expense', 'Distance from Residence to Wor
k',
       'Service time', 'Age', 'Work load Average/day ', 'Hit target',
       'Disciplinary failure', 'Education', 'Son', 'Social drinker',
       'Social smoker', 'Pet', 'Weight', 'Height', 'Body mass index',
       'Absenteeism time in hours'],
      dtype='object')
```

In [118…
```
count = pd.DataFrame(absent['ID'].value_counts())

count = count.sort_values('ID')

print(count) #IDs 8, 4, and 35 have no hours of absence, but this will be ta
```

```
    count
ID
1      23
2       6
3     113
4       1
5      19
6       8
7       6
8       2
9       8
10     24
11     40
12      7
13     15
14     29
15     37
16      2
17     20
18     16
19      3
20     42
21      3
22     46
23      8
24     30
25     10
26      5
27      7
28     76
29      5
30      7
31      3
32      5
33     24
34     55
35      1
36     34
```

In [119…
```
count_reason = pd.DataFrame(absent['Reason for absence'].value_counts())
```

```
count_reason = count_reason.sort_values('count', ascending=True)

print(count_reason)
```

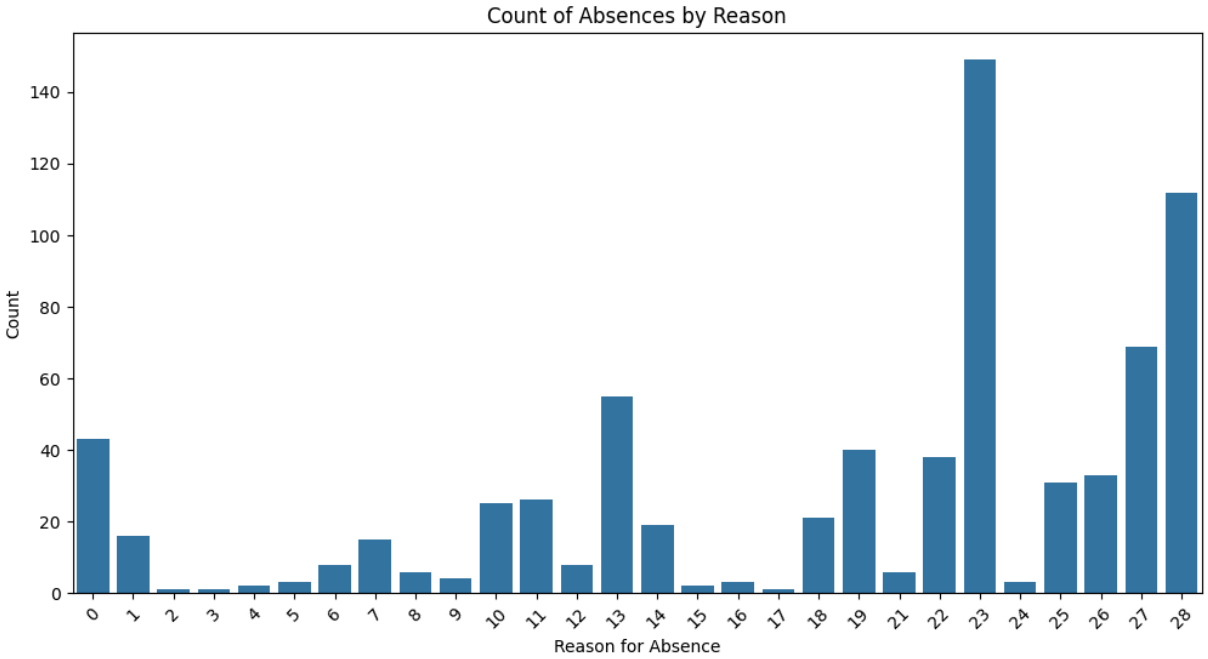
```
                    count
Reason for absence
17                      1
3                       1
2                       1
4                       2
15                      2
24                      3
16                      3
5                       3
9                       4
8                       6
21                      6
12                      8
6                       8
7                      15
1                      16
14                     19
18                     21
10                     25
11                     26
25                     31
26                     33
22                     38
19                     40
0                      43
13                     55
27                     69
28                    112
23                    149
```

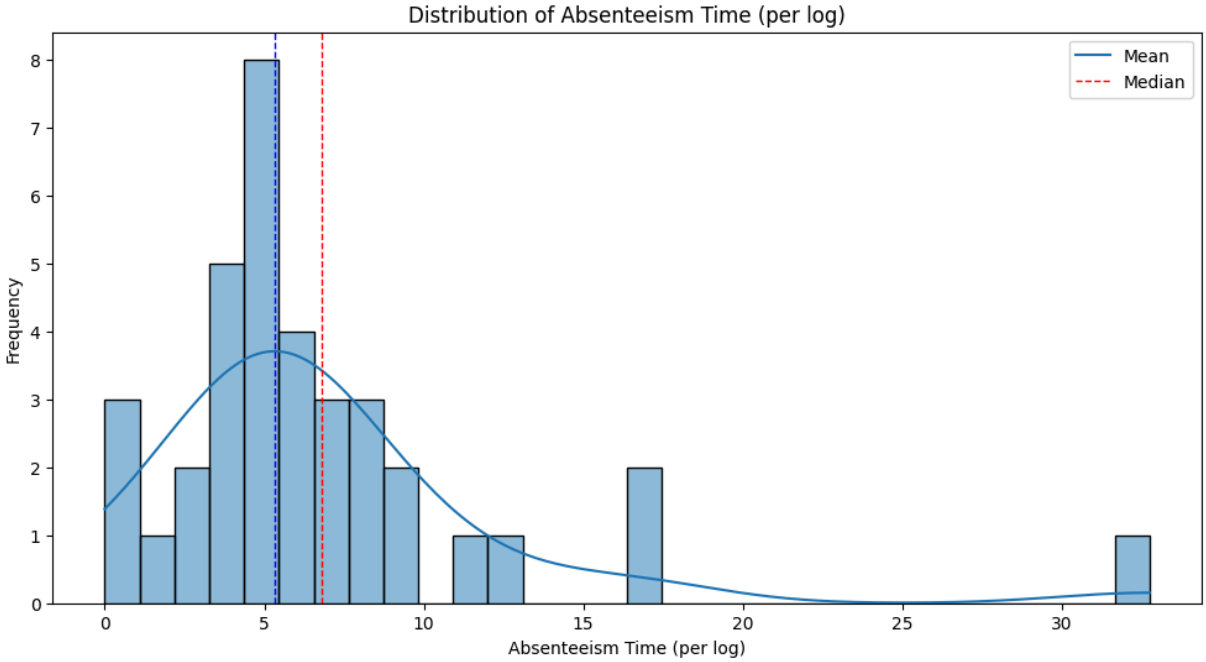
In [120…
```
indv_info = pd.DataFrame(absent.groupby(['ID'])['Absenteeism time in hours']

indv_info['abs_hr_per_month'] = indv_info['Absenteeism time in hours']/36

indv_info['abs_hr_per_log'] = indv_info['Absenteeism time in hours']/count['
```

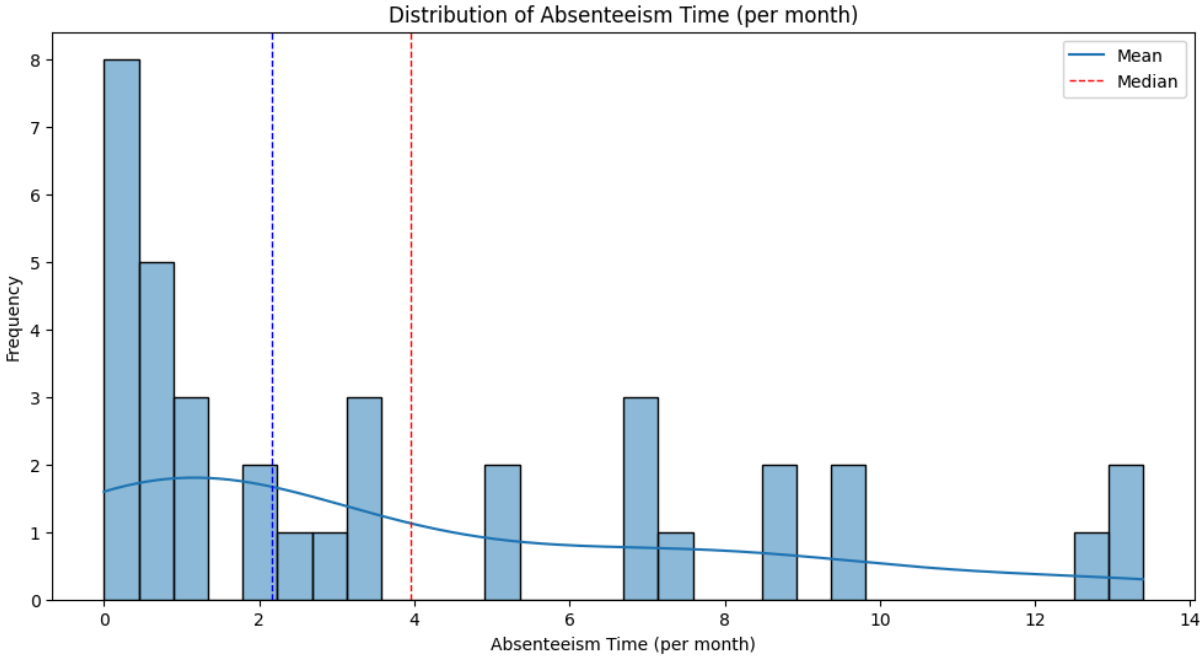
EDA Plots

In [121…
```
plt.figure(figsize=(12, 6))
sns.countplot(data=absent, x='Reason for absence')
plt.title('Count of Absences by Reason')
plt.xlabel('Reason for Absence')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(12, 6))
sns.histplot(indv_info['abs_hr_per_log'], bins=30, kde=True)
plt.title('Distribution of Absenteeism Time (per log)')
plt.xlabel('Absenteeism Time (per log)')
```

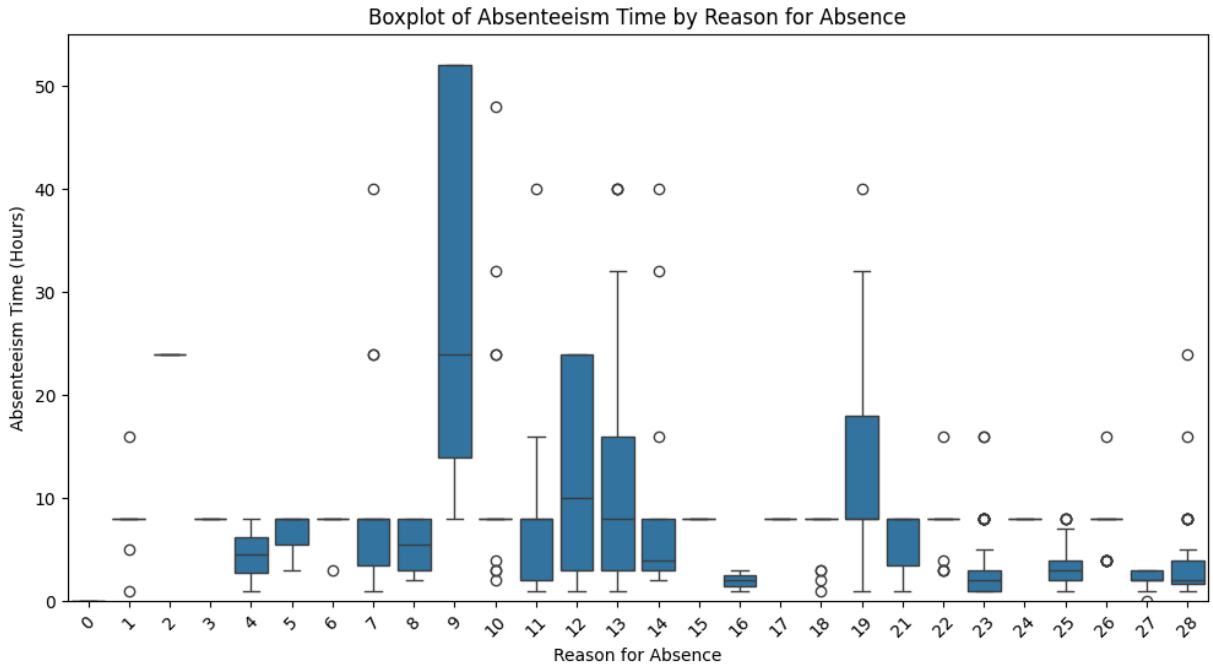
```
plt.ylabel('Frequency')
plt.axvline(indv_info['abs_hr_per_log'].mean(), color='red', linestyle='dash
plt.axvline(indv_info['abs_hr_per_log'].median(), color='blue', linestyle='d
plt.legend({'Mean': indv_info['abs_hr_per_log'].mean(), 'Median': indv_info[
plt.show()

plt.figure(figsize=(12, 6))
sns.histplot(indv_info['abs_hr_per_month'], bins=30, kde=True)
plt.title('Distribution of Absenteeism Time (per month)')
plt.xlabel('Absenteeism Time (per month)')
plt.ylabel('Frequency')
plt.axvline(indv_info['abs_hr_per_month'].mean(), color='red', linestyle='da
plt.axvline(indv_info['abs_hr_per_month'].median(), color='blue', linestyle=
plt.legend({'Mean': indv_info['abs_hr_per_month'].mean(), 'Median': indv_inf
plt.show()

plt.figure(figsize=(12, 6))
sns.boxplot(absent, x='Reason for absence', y='Absenteeism time in hours')
plt.title('Boxplot of Absenteeism Time by Reason for Absence')
plt.xlabel('Reason for Absence')
plt.ylabel('Absenteeism Time (Hours)')
plt.xticks(rotation=45)
plt.ylim(top=55, bottom=0)
plt.show()

plt.figure(figsize=(12, 6))
sns.boxplot(absent, x='Month of absence', y='Absenteeism time in hours')
plt.title('Boxplot of Absenteeism Time by Month')
plt.xlabel('Month of Absence')
plt.ylabel('Absenteeism Time (Hours)')
plt.ylim(top=30, bottom=0)
plt.xticks(rotation=45)
plt.show()
```



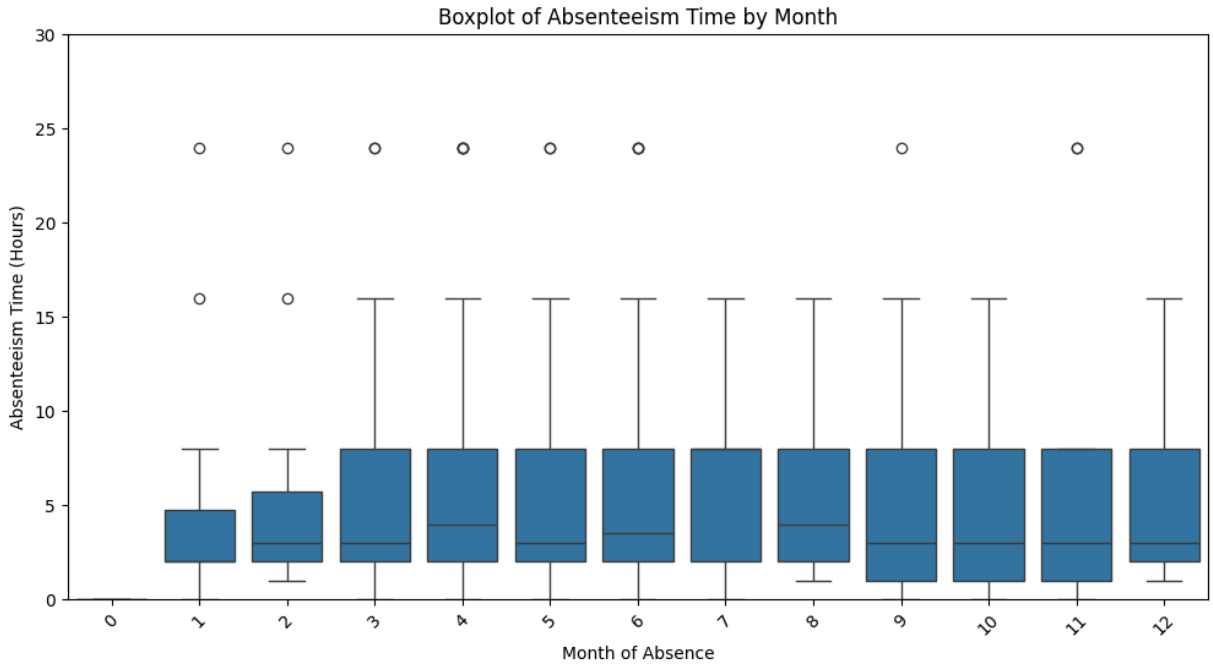





```
/Users/patriciomartinez/miniconda3/lib/python3.11/site-packages/seaborn/categorical.py:632: FutureWarning: SeriesGroupBy.grouper is deprecated and will be removed in a future version of pandas.
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```




```
/Users/patriciomartinez/miniconda3/lib/python3.11/site-packages/seaborn/categorical.py:632: FutureWarning: SeriesGroupBy.grouper is deprecated and will be removed in a future version of pandas.
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```



In [122…
```
print(indv_info.sort_values('Absenteeism time in hours', ascending=False))
```

ID	Absenteeism time in hours	abs_hr_per_month	abs_hr_per_log
3	482	13.388889	4.265487
14	476	13.222222	16.413793
11	450	12.500000	11.250000
28	347	9.638889	4.565789
34	344	9.555556	6.254545
36	311	8.638889	9.147059
20	306	8.500000	7.285714
9	262	7.277778	32.750000
24	254	7.055556	8.466667
15	253	7.027778	6.837838
22	253	7.027778	5.500000
10	186	5.166667	7.750000
13	183	5.083333	12.200000
17	126	3.500000	6.300000
1	121	3.361111	5.260870
18	118	3.277778	7.375000
5	104	2.888889	5.473684
26	83	2.305556	16.600000
33	73	2.027778	3.041667
6	72	2.000000	9.000000
25	42	1.166667	4.200000
23	40	1.111111	5.000000
12	34	0.944444	4.857143
30	31	0.861111	4.428571
7	30	0.833333	5.000000
27	27	0.750000	3.857143
2	25	0.694444	4.166667
29	21	0.583333	4.200000
31	16	0.444444	5.333333
32	16	0.444444	3.200000
16	16	0.444444	8.000000
21	16	0.444444	5.333333
19	6	0.166667	2.000000
8	0	0.000000	0.000000
4	0	0.000000	0.000000
35	0	0.000000	0.000000

In [123…
```
print(indv_info.sort_values('abs_hr_per_month', ascending=False))
```

ID	Absenteeism time in hours	abs_hr_per_month	abs_hr_per_log
3	482	13.388889	4.265487
14	476	13.222222	16.413793
11	450	12.500000	11.250000
28	347	9.638889	4.565789
34	344	9.555556	6.254545
36	311	8.638889	9.147059
20	306	8.500000	7.285714
9	262	7.277778	32.750000
24	254	7.055556	8.466667
15	253	7.027778	6.837838
22	253	7.027778	5.500000
10	186	5.166667	7.750000
13	183	5.083333	12.200000
17	126	3.500000	6.300000
1	121	3.361111	5.260870
18	118	3.277778	7.375000
5	104	2.888889	5.473684
26	83	2.305556	16.600000
33	73	2.027778	3.041667
6	72	2.000000	9.000000
25	42	1.166667	4.200000
23	40	1.111111	5.000000
12	34	0.944444	4.857143
30	31	0.861111	4.428571
7	30	0.833333	5.000000
27	27	0.750000	3.857143
2	25	0.694444	4.166667
29	21	0.583333	4.200000
31	16	0.444444	5.333333
32	16	0.444444	3.200000
16	16	0.444444	8.000000
21	16	0.444444	5.333333
19	6	0.166667	2.000000
8	0	0.000000	0.000000
4	0	0.000000	0.000000
35	0	0.000000	0.000000

In [124…
```
print(pd.DataFrame(absent[(absent['ID'] == 9)].value_counts('Reason for abse
```

Reason for absence	count
6	2
18	2
25	2
1	1
12	1

In [125…
```
print(pd.DataFrame(absent[(absent['ID'] == 3)].value_counts('Reason for abse
```

Reason for absence	count
27	38
28	26
23	19
13	10
11	7
10	2
18	2
21	2
25	2
0	1
5	1
6	1
12	1
26	1

In [126…
```
print(pd.DataFrame(indv_info[(indv_info['Absenteeism time in hours'] < 20)])
```

ID	Absenteeism time in hours	abs_hr_per_month	abs_hr_per_log
4	0	0.000000	0.000000
8	0	0.000000	0.000000
16	16	0.444444	8.000000
19	6	0.166667	2.000000
21	16	0.444444	5.333333
31	16	0.444444	5.333333
32	16	0.444444	3.200000
35	0	0.000000	0.000000

Data Processing

In [127…
```
absent.columns = ['worker_id', 'absence_reason', 'absence_month', 'absence_c

absent.dtypes
```

Out[127…
```
worker_id              int64
absence_reason         int64
absence_month          int64
absence_day            int64
season                 int64
trans_exp              int64
distance_to_work       int64
serve_time             int64
Age                    int64
avg_work_per_day     float64
target_hit             int64
disc_fail              int64
educ                   int64
children               int64
soc_drink              int64
soc_smoke              int64
pets                   int64
weight                 int64
height                 int64
BMI                    int64
absence_hours          int64
dtype: object
```

In [128…
```
print(stats.median(absent['absence_hours']))
print(stats.stdev(absent['absence_hours']))
print(stats.median(absent['absence_hours']) + 2*stats.stdev(absent['absence_
```

```
3.0
13.330998100978201
29.661996201956402
```

In [129…
```
cutoff = stats.median(absent['absence_hours']) + 2*stats.stdev(absent['abser
absent['above_normal'] = absent['absence_hours'] > cutoff
```

In [130…
```
absent['absence_month'] = absent['absence_month'].replace([1, 2, 3, 4, 5, 6,
absent['absence_day'] = absent['absence_day'].replace([1, 2, 3, 4, 5, 6, 7],
absent['season'] = absent['season'].replace([1, 2, 3, 4], ['summer', 'autumr
absent['disc_fail'] = absent['disc_fail'].replace([0,1], ['no', 'yes'])
absent['soc_drink'] = absent['soc_drink'].replace([0,1], ['no', 'yes'])
absent['soc_smoke'] = absent['soc_smoke'].replace([0,1], ['no', 'yes'])

absent['absence_reason'] = absent['absence_reason'].astype('object')
```

In [131…
```
X= absent.drop(columns = ['absence_hours', 'worker_id', 'absence_hours', 'ab
y = absent['above_normal']
```

In [132…
```
dum_col = ['absence_reason', 'absence_month', 'absence_day', 'season', 'disc
X_dum = pd.get_dummies(X[dum_col], drop_first=True)
X_dum.head()
```

Out[132…

	absence_reason_1	absence_reason_2	absence_reason_3	absence_reason_4	abse
0	False	False	False	False	
1	False	False	False	False	
2	False	False	False	False	
3	False	False	False	False	
4	False	False	False	False	

5 rows × 49 columns

In [133…
```
scaler = preprocessing.StandardScaler()
X_norm = X.drop(columns = dum_col)
X_norm = pd.DataFrame(scaler.fit_transform(X_norm*1.0), columns=X_norm.colum
```

In [134…
```
X_new = pd.concat([X_dum, X_norm], axis = 1)

ros = RandomOverSampler(random_state=1)
X_res, y_res = ros.fit_resample(X_new, y)
```

In [135…
```
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=
```

In [136…
```
predictors = X_train.columns
X_train
```

Out[136…

	absence_reason_1	absence_reason_2	absence_reason_3	absence_reason_4	a
792	False	False	False	False	
1067	False	False	False	False	
652	False	False	False	False	
1286	False	False	False	False	
803	False	False	False	False	
…	…	…	…	…	
1381	False	False	False	False	
382	False	False	False	False	
129	False	False	False	False	
1309	False	False	False	False	
482	False	False	False	False	

854 rows × 60 columns

Logistic Regression

In [137…

```
pd.set_option('display.max_rows', 62)

log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)

predictors = X_train.columns
coef = log_reg.coef_.reshape(60,)

coef = pd.DataFrame({'Predictor': predictors, 'coefficient': coef})
coef['exp'] = exp(coef['coefficient'])
display(coef.sort_values(by = ['exp'], ascending=False))

log_pred = log_reg.predict(X_test)

print(f'accuracy: {accuracy_score(y_test, log_pred)}')
print(f'precision: {precision_score(y_test, log_pred, zero_division=0)}')
print(f'recall: {recall_score(y_test, log_pred, zero_division=1)}')
print('\n')

classificationSummary(y_test, log_pred)
```

	Predictor	coefficient	exp
8	absence_reason_9	3.630728	37.740282
18	absence_reason_19	3.033236	20.764319
12	absence_reason_13	2.769181	15.945562
0	absence_reason_1	1.718811	5.577895
39	absence_day_Mon	1.665998	5.290949
13	absence_reason_14	1.260764	3.528115
43	season_spring	1.242336	3.463695
29	absence_month_Dec	1.096538	2.993783
32	absence_month_Jul	1.087845	2.967872
10	absence_reason_11	0.999778	2.717679
59	height	0.973989	2.648488
28	absence_month_Aug	0.706330	2.026539
52	Age	0.668355	1.951024
11	absence_reason_12	0.577494	1.781569
56	children	0.427623	1.533608
41	absence_day_Tue	0.422163	1.525258
38	absence_month_Sep	0.382859	1.466472
5	absence_reason_6	0.288850	1.334891
47	soc_drink_yes	0.240438	1.271806
42	absence_day_Wed	0.233467	1.262972
6	absence_reason_7	0.215532	1.240522
33	absence_month_Jun	0.160469	1.174061
9	absence_reason_10	0.044416	1.045418
14	absence_reason_15	0.000000	1.000000
49	trans_exp	-0.045311	0.955701
27	absence_month_Apr	-0.057104	0.944495
48	soc_smoke_yes	-0.067255	0.934957
54	target_hit	-0.082883	0.920459
2	absence_reason_3	-0.095008	0.909366
53	avg_work_per_day	-0.098368	0.906316
34	absence_month_Mar	-0.104776	0.900526
37	absence_month_Oct	-0.107056	0.898476

	Predictor	coefficient	exp
50	distance_to_work	-0.110782	0.895134
1	absence_reason_2	-0.117840	0.888838
3	absence_reason_4	-0.155502	0.855985
51	serve_time	-0.167070	0.846140
4	absence_reason_5	-0.170371	0.843352
16	absence_reason_17	-0.174676	0.839729
15	absence_reason_16	-0.243344	0.784002
19	absence_reason_21	-0.297633	0.742574
31	absence_month_Jan	-0.360033	0.697653
35	absence_month_May	-0.360795	0.697122
45	season_winter	-0.366341	0.693267
57	pets	-0.393507	0.674687
7	absence_reason_8	-0.409170	0.664201
22	absence_reason_24	-0.426108	0.653046
55	educ	-0.428911	0.651218
17	absence_reason_18	-0.490051	0.612595
44	season_summer	-0.673290	0.510028
25	absence_reason_27	-0.766028	0.464856
36	absence_month_Nov	-0.809456	0.445100
58	weight	-0.934451	0.392801
20	absence_reason_22	-1.340962	0.261594
46	disc_fail_yes	-1.469961	0.229934
24	absence_reason_26	-1.548821	0.212498
30	absence_month_Feb	-1.610201	0.199847
23	absence_reason_25	-1.649714	0.192105
40	absence_day_Thu	-2.453531	0.085989
21	absence_reason_23	-2.454191	0.085933
26	absence_reason_28	-2.704789	0.066884

```
accuracy: 0.9438596491228071
precision: 0.8990536277602523
recall: 1.0


Confusion Matrix (Accuracy 0.9439)

       Prediction
Actual   0   1
     0 253  32
     1   0 285
```

Random Forest (with hyperparameter tuning)

In [138…

```
grid_space = {'max_depth': [5, 10, 15],
              'n_estimators': [5, 10, 15, 20, 25, 30]}

rf = RandomForestClassifier(random_state=12345)

grid = GridSearchCV(rf, param_grid = grid_space, cv = 5, scoring = 'accuracy
rf_fit = grid.fit(X_train, y_train)

print('Best hyperparameters are: '+str(rf_fit.best_params_))
print('Best score is: '+str(rf_fit.best_score_))


rf_hp = RandomForestClassifier(max_depth=15, n_estimators=10, random_state=1
rf_fit = rf_hp.fit(X_train, y_train)

rf_pred = rf_fit.predict(X_test)

print(f'accuracy: {accuracy_score(y_test, rf_pred)}')
print(f'precision: {precision_score(y_test, rf_pred, zero_division=0)}')
print(f'recall: {recall_score(y_test, rf_pred, zero_division=1)}')
print('\n')

print(classificationSummary(y_test, rf_pred))

importances = rf_fit.feature_importances_
feature_imp_df = pd.DataFrame({'Feature': predictors, 'Gini Importance': imp
display(feature_imp_df)
```

```
Best hyperparameters are: {'max_depth': 15, 'n_estimators': 10}
Best score is: 0.9859442724458203
accuracy: 0.9929824561403509
precision: 0.986159169550173
recall: 1.0


Confusion Matrix (Accuracy 0.9930)

       Prediction
Actual   0   1
     0 281   4
     1   0 285
None
```

	Feature	Gini Importance
21	absence_reason_23	0.066914
53	avg_work_per_day	0.065565
50	distance_to_work	0.064968
49	trans_exp	0.058090
40	absence_day_Thu	0.051818
59	height	0.049804
18	absence_reason_19	0.047863
54	target_hit	0.042761
12	absence_reason_13	0.041536
39	absence_day_Mon	0.040088
56	children	0.039325
26	absence_reason_28	0.038190
52	Age	0.028421
51	serve_time	0.028111
8	absence_reason_9	0.027652
37	absence_month_Oct	0.019953
58	weight	0.018867
23	absence_reason_25	0.017234
42	absence_day_Wed	0.015811
41	absence_day_Tue	0.015217
57	pets	0.014452
10	absence_reason_11	0.014083
25	absence_reason_27	0.013197
55	educ	0.012065
29	absence_month_Dec	0.011459
46	disc_fail_yes	0.011411
30	absence_month_Feb	0.011355
24	absence_reason_26	0.010917
36	absence_month_Nov	0.010532
13	absence_reason_14	0.010432
0	absence_reason_1	0.010387
32	absence_month_Jul	0.008525

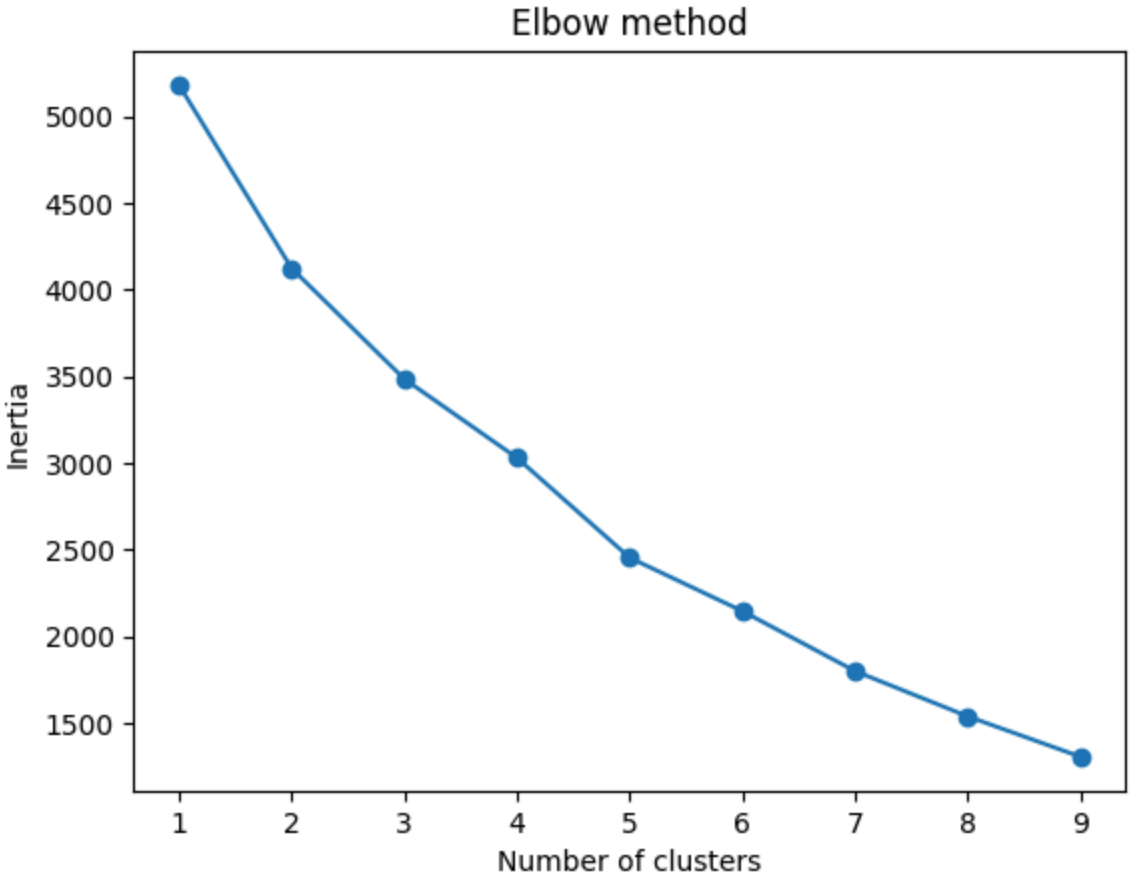
	Feature	Gini Importance
47	soc_drink_yes	0.007598
5	absence_reason_6	0.007369
44	season_summer	0.006964
35	absence_month_May	0.006369
27	absence_month_Apr	0.005937
31	absence_month_Jan	0.005905
20	absence_reason_22	0.005520
34	absence_month_Mar	0.005487
9	absence_reason_10	0.005235
11	absence_reason_12	0.005032
6	absence_reason_7	0.004471
17	absence_reason_18	0.003321
45	season_winter	0.002971
48	soc_smoke_yes	0.002744
28	absence_month_Aug	0.002421
43	season_spring	0.001908
4	absence_reason_5	0.001005
7	absence_reason_8	0.000889
38	absence_month_Sep	0.000795
33	absence_month_Jun	0.000581
16	absence_reason_17	0.000475
14	absence_reason_15	0.000000
15	absence_reason_16	0.000000
19	absence_reason_21	0.000000
1	absence_reason_2	0.000000
3	absence_reason_4	0.000000
2	absence_reason_3	0.000000
22	absence_reason_24	0.000000

K-Means Clustering

In [139…
```
inertias = []
```

```
for i in range(1,10):
    kmeans = KMeans(n_clusters=i, random_state=12345)
    kmeans.fit(X_new[['distance_to_work', 'trans_exp', 'height', 'children',
    inertias.append(kmeans.inertia_)

plt.plot(range(1,10), inertias, marker='o')
plt.title('Elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()

kmeans = KMeans(n_clusters=4)
kmeans_absent = kmeans.fit(X_new[['distance_to_work', 'trans_exp', 'height',

pd.DataFrame(kmeans_absent.cluster_centers_, columns = ['distance_to_work',
```

```
/Users/patriciomartinez/miniconda3/lib/python3.11/site-packages/sklearn/clus
ter/_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/patriciomartinez/miniconda3/lib/python3.11/site-packages/sklearn/clus
ter/_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/patriciomartinez/miniconda3/lib/python3.11/site-packages/sklearn/clus
ter/_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/patriciomartinez/miniconda3/lib/python3.11/site-packages/sklearn/clus
ter/_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/patriciomartinez/miniconda3/lib/python3.11/site-packages/sklearn/clus
ter/_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/patriciomartinez/miniconda3/lib/python3.11/site-packages/sklearn/clus
ter/_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/patriciomartinez/miniconda3/lib/python3.11/site-packages/sklearn/clus
ter/_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/patriciomartinez/miniconda3/lib/python3.11/site-packages/sklearn/clus
ter/_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/patriciomartinez/miniconda3/lib/python3.11/site-packages/sklearn/clus
ter/_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
```




```
/Users/patriciomartinez/miniconda3/lib/python3.11/site-packages/sklearn/clus
ter/_kmeans.py:1416: FutureWarning: The default value of `n_init` will chang
e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

Out[139…

	distance_to_work	trans_exp	height	children	Age	weight	pet
0	-1.095044	-0.632301	0.949945	-0.279412	0.295912	0.794839	-0.45145
1	1.429338	-0.563587	-0.329770	-0.920536	0.330260	0.828175	-0.49607
2	-0.002825	0.343115	-0.418788	0.433320	-0.198495	-0.687090	0.02461
3	1.205180	1.516003	-0.011837	0.141193	-0.576519	-0.158075	3.09719

Lasso Regression

In [140…

```
from sklearn.metrics import classification_report

# Lasso (L1) Logistic Regression
lasso_params = {'C': [0.01, 0.1, 1, 10, 85, 100]}
lasso_clf = GridSearchCV(LogisticRegression(penalty='l1', solver='liblinear'
lasso_clf.fit(X_train, y_train)

# Best parameters and evaluation for Lasso
print("\nBest C for Lasso (L1):", lasso_clf.best_params_['C'])
lasso_best = lasso_clf.best_estimator_
```

```
lasso_pred = lasso_best.predict(X_test)

# Evaluate Lasso (L1) Classification
print("\nLasso (L1) Logistic Regression Evaluation:")
print(f'Accuracy: {accuracy_score(y_test, lasso_pred)}')
print(f'Precision: {precision_score(y_test, lasso_pred, zero_division=0)}')
print(f'Recall: {recall_score(y_test, lasso_pred, zero_division=0)}')
print(f'F1 Score: {f1_score(y_test, lasso_pred)}')
print(classificationSummary(y_test, lasso_pred))
print("\nClassification Report for Lasso (L1):")
print(classification_report(y_test, lasso_pred))
```

```
Best C for Lasso (L1): 85

Lasso (L1) Logistic Regression Evaluation:
Accuracy: 0.9491228070175438
Precision: 0.9076433121019108
Recall: 1.0
F1 Score: 0.9515859766277129
Confusion Matrix (Accuracy 0.9491)

       Prediction
Actual   0   1
     0 256  29
     1   0 285
None

Classification Report for Lasso (L1):
              precision    recall  f1-score   support

       False       1.00      0.90      0.95       285
        True       0.91      1.00      0.95       285

    accuracy                           0.95       570
   macro avg       0.95      0.95      0.95       570
weighted avg       0.95      0.95      0.95       570
```

In [141…

```
predictors = X_train.columns
coef = lasso_best.coef_.reshape(60,)

coef = pd.DataFrame({'Predictor': predictors, 'coefficient': coef})
coef['exp'] = exp(coef['coefficient'])
display(coef.sort_values(by = ['exp'], ascending=False))
```

	Predictor	coefficient	exp
8	absence_reason_9	24.588941	4.773549e+10
28	absence_month_Aug	10.450206	3.455149e+04
18	absence_reason_19	7.971493	2.897179e+03
10	absence_reason_11	7.698733	2.205551e+03
12	absence_reason_13	7.529969	1.863049e+03
13	absence_reason_14	7.333340	1.530486e+03
32	absence_month_Jul	6.619208	7.493515e+02
43	season_spring	5.716155	3.037347e+02
0	absence_reason_1	4.500000	9.001712e+01
5	absence_reason_6	4.342350	7.688804e+01
59	height	3.503113	3.321870e+01
39	absence_day_Mon	3.315896	2.754707e+01
52	Age	2.671904	1.446750e+01
38	absence_month_Sep	2.242493	9.416774e+00
54	target_hit	1.818720	6.163964e+00
56	children	1.733252	5.659026e+00
49	trans_exp	1.400607	4.057664e+00
6	absence_reason_7	1.135517	3.112782e+00
33	absence_month_Jun	0.826739	2.285852e+00
11	absence_reason_12	0.490564	1.633237e+00
50	distance_to_work	0.369192	1.446566e+00
19	absence_reason_21	0.000000	1.000000e+00
2	absence_reason_3	0.000000	1.000000e+00
16	absence_reason_17	0.000000	1.000000e+00
15	absence_reason_16	0.000000	1.000000e+00
14	absence_reason_15	0.000000	1.000000e+00
4	absence_reason_5	0.000000	1.000000e+00
7	absence_reason_8	0.000000	1.000000e+00
37	absence_month_Oct	0.000000	1.000000e+00
3	absence_reason_4	0.000000	1.000000e+00
42	absence_day_Wed	-0.030833	9.696375e-01
55	educ	-0.223406	7.997899e-01

	Predictor	coefficient	exp
53	avg_work_per_day	-0.649265	5.224296e-01
29	absence_month_Dec	-0.826161	4.377266e-01
48	soc_smoke_yes	-0.875778	4.165379e-01
9	absence_reason_10	-1.059055	3.467833e-01
58	weight	-1.165237	3.118488e-01
51	serve_time	-1.367021	2.548651e-01
1	absence_reason_2	-1.439527	2.370398e-01
47	soc_drink_yes	-1.560746	2.099794e-01
41	absence_day_Tue	-1.713555	1.802240e-01
57	pets	-1.876804	1.530786e-01
45	season_winter	-2.619752	7.282094e-02
17	absence_reason_18	-2.942757	5.272020e-02
27	absence_month_Apr	-3.248754	3.882256e-02
31	absence_month_Jan	-4.455580	1.161358e-02
25	absence_reason_27	-4.497523	1.113655e-02
34	absence_month_Mar	-5.325627	4.865302e-03
22	absence_reason_24	-5.420574	4.424607e-03
35	absence_month_May	-6.065128	2.322462e-03
24	absence_reason_26	-6.771198	1.146320e-03
46	disc_fail_yes	-8.757770	1.572349e-04
23	absence_reason_25	-8.763355	1.563591e-04
20	absence_reason_22	-9.190750	1.019784e-04
36	absence_month_Nov	-9.935889	4.840587e-05
21	absence_reason_23	-10.896977	1.851412e-05
26	absence_reason_28	-12.299696	4.553127e-06
44	season_summer	-12.451472	3.911960e-06
30	absence_month_Feb	-18.242115	1.195502e-08
40	absence_day_Thu	-18.815407	6.738643e-09

Ridge Regression

In [142…

```
# Ridge (L2) Logistic Regression
ridge_params = {'C': [0.01, 0.1, 1, 10, 100]}
```

```
ridge_clf = GridSearchCV(LogisticRegression(penalty='l2', solver='liblinear'
ridge_clf.fit(X_train, y_train)

# Best parameters and evaluation for Ridge
print("\nBest C for Ridge (L2):", ridge_clf.best_params_['C'])
ridge_best = ridge_clf.best_estimator_
ridge_pred = ridge_best.predict(X_test)

# Evaluate Ridge (L2) Classification
print("\nRidge (L2) Logistic Regression Evaluation:")
print(f'Accuracy: {accuracy_score(y_test, ridge_pred)}')
print(f'Precision: {precision_score(y_test, ridge_pred, zero_division=0)}')
print(f'Recall: {recall_score(y_test, ridge_pred, zero_division=0)}')
print(f'F1 Score: {f1_score(y_test, ridge_pred)}')
print(classificationSummary(y_test, ridge_pred))
print("\nClassification Report for Ridge (L2):")
print(classification_report(y_test, ridge_pred))
```

```
Best C for Ridge (L2): 100

Ridge (L2) Logistic Regression Evaluation:
Accuracy: 0.9491228070175438
Precision: 0.9076433121019108
Recall: 1.0
F1 Score: 0.9515859766277129
Confusion Matrix (Accuracy 0.9491)

       Prediction
Actual   0   1
     0 256  29
     1   0 285
None

Classification Report for Ridge (L2):
              precision    recall  f1-score   support

       False       1.00      0.90      0.95       285
        True       0.91      1.00      0.95       285

    accuracy                           0.95       570
   macro avg       0.95      0.95      0.95       570
weighted avg       0.95      0.95      0.95       570
```

In [143…

```
predictors = X_train.columns
coef = ridge_best.coef_.reshape(60,)

coef = pd.DataFrame({'Predictor': predictors, 'coefficient': coef})
coef['exp'] = exp(coef['coefficient'])
display(coef.sort_values(by = ['exp'], ascending=False))
```

	Predictor	coefficient	exp
8	absence_reason_9	16.953346	2.305391e+07
28	absence_month_Aug	7.748964	2.319170e+03
18	absence_reason_19	6.672711	7.905361e+02
12	absence_reason_13	6.316945	5.538782e+02
13	absence_reason_14	5.628223	2.781674e+02
10	absence_reason_11	5.615326	2.746029e+02
32	absence_month_Jul	4.787966	1.200569e+02
43	season_spring	3.976848	5.334863e+01
0	absence_reason_1	3.911930	4.999534e+01
38	absence_month_Sep	3.536046	3.433092e+01
5	absence_reason_6	3.512737	3.353994e+01
39	absence_day_Mon	2.962117	1.933886e+01
59	height	2.741690	1.551318e+01
52	Age	2.039358	7.685670e+00
33	absence_month_Jun	1.743153	5.715333e+00
37	absence_month_Oct	1.739199	5.692779e+00
29	absence_month_Dec	1.535263	4.642547e+00
56	children	1.424074	4.154010e+00
54	target_hit	1.257032	3.514974e+00
11	absence_reason_12	0.959008	2.609106e+00
49	trans_exp	0.914256	2.494919e+00
6	absence_reason_7	0.871493	2.390477e+00
50	distance_to_work	0.242377	1.274275e+00
42	absence_day_Wed	0.017011	1.017156e+00
14	absence_reason_15	0.000000	1.000000e+00
3	absence_reason_4	-0.004734	9.952772e-01
16	absence_reason_17	-0.052071	9.492618e-01
4	absence_reason_5	-0.130360	8.777794e-01
15	absence_reason_16	-0.144790	8.652038e-01
19	absence_reason_21	-0.213735	8.075623e-01
55	educ	-0.288897	7.490895e-01
53	avg_work_per_day	-0.446583	6.398106e-01

	Predictor	coefficient	exp
2	absence_reason_3	-0.511498	5.995968e-01
48	soc_smoke_yes	-0.558642	5.719851e-01
9	absence_reason_10	-0.750834	4.719726e-01
7	absence_reason_8	-0.859669	4.233022e-01
51	serve_time	-0.891754	4.099360e-01
58	weight	-1.071195	3.425989e-01
27	absence_month_Apr	-1.113196	3.285075e-01
47	soc_drink_yes	-1.154649	3.151681e-01
41	absence_day_Tue	-1.210352	2.980925e-01
1	absence_reason_2	-1.284907	2.766762e-01
57	pets	-1.363596	2.557395e-01
45	season_winter	-1.988807	1.368586e-01
31	absence_month_Jan	-2.217258	1.089073e-01
17	absence_reason_18	-2.561721	7.717181e-02
34	absence_month_Mar	-2.730749	6.517044e-02
35	absence_month_May	-3.034502	4.809863e-02
25	absence_reason_27	-3.388993	3.374265e-02
22	absence_reason_24	-4.039582	1.760483e-02
24	absence_reason_26	-5.239522	5.302792e-03
36	absence_month_Nov	-5.592527	3.725602e-03
23	absence_reason_25	-6.204492	2.020335e-03
46	disc_fail_yes	-6.241806	1.946337e-03
20	absence_reason_22	-6.544030	1.438679e-03
44	season_summer	-7.744393	4.331643e-04
21	absence_reason_23	-7.954083	3.512252e-04
26	absence_reason_28	-8.814118	1.486200e-04
30	absence_month_Feb	-10.901585	1.842900e-05
40	absence_day_Thu	-13.338579	1.611122e-06