ML_MP4

Clarice Tee

2025-03-14

1.

```
# Importing necessary libraries from scikit-learn for Support Vector Machine (SVM) classi
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, KFold
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

# Importing essential libraries for data manipulation and visualization
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os

# Importing datasets module from scikit-learn for access to built-in datasets
from sklearn import datasets
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score

from sklearn import datasets
```

Data types: vote and work are integers (binary)

prtage is numerical (int)

Categorical variables (pesex, ptdtrace, pehspnon, prcitshp, peeduca, vote) are encoded as dummy variables

```
directory = r"C:\Users\clari\OneDrive\Documents\Machine Learning\mp4"
df_vote = os.path.join(directory, "vote.csv")
df_vote = pd.read_csv(df_vote, encoding="latin1")
print(df_vote.shape, "\n")
```

```
print(df_vote.dtypes)
print(df_vote.shape)
df_vote.head()
```

```
(5000, 7)

prtage       int64
pesex       object
ptdtrace    object
pehspnon    object
prcitshp    object
peeduca     object
vote        object
dtype: object
(5000, 7)
```

	prtage	pesex	ptdtrace	pehspnon	prcitshp	peeduc
0	19	FEMALE	White Only	NON-HISPANIC	NATIVE, BORN IN THE UNITED	SOME
1	35	MALE	White Only	NON-HISPANIC	NATIVE, BORN IN THE UNITED	MAST
2	48	MALE	White Only	HISPANIC	FOREIGN BORN, U.S. CITIZEN BY	5TH C
3	55	MALE	White Only	NON-HISPANIC	NATIVE, BORN IN THE UNITED	BACH
4	25	FEMALE	White Only	NON-HISPANIC	NATIVE, BORN IN THE UNITED	SOME

```
directory = r"C:\Users\clari\OneDrive\Documents\Machine Learning\mp4"
df_work = os.path.join(directory, "work.csv")
df_work = pd.read_csv(df_work, encoding="latin1")
print(df_work.shape, "\n")
print(df_work.dtypes)
```

```
(5000, 7)

prtage       int64
pesex       object
ptdtrace    object
pehspnon    object
prcitshp    object
peeduca     object
work        object
dtype: object
```

```
na_count_vote = pd.DataFrame(np.sum(df_vote.isna(), axis = 0), columns = ["Count NAs"])
print(na_count_vote)

na_count_work = pd.DataFrame(np.sum(df_work.isna(), axis = 0), columns = ["Count NAs"])
print(na_count_work)
```

```
          Count NAs
prtage            0
pesex             0
ptdtrace          0
pehspnon          0
prcitshp          0
peeduca           0
vote              0
          Count NAs
prtage            0
pesex             0
ptdtrace          0
pehspnon          0
prcitshp          0
peeduca           0
work              0
```

2.

```
# Map labels to binary
work_mapper = {'flexible': 1, "not flexible": 0}
vote_mapper = {'vote': 1, "did not vote": 0}
df_work['work'] = df_work['work'].replace(work_mapper)
df_vote['vote'] = df_vote['vote'].replace(vote_mapper)
```

```
C:\Users\clari\AppData\Local\Temp\ipykernel_22776\1992466733.py:4: FutureWarning:

Downcasting behavior in `replace` is deprecated and will be removed in a future version.

C:\Users\clari\AppData\Local\Temp\ipykernel_22776\1992466733.py:5: FutureWarning:

Downcasting behavior in `replace` is deprecated and will be removed in a future version.
```

2.c.and 2.d.

```
# One-hot encoding for categorical variables
df_work = pd.get_dummies(df_work, columns=['ptdtrace', 'pesex', 'pehspnon', 'peeduca', '
df_vote = pd.get_dummies(df_vote, columns=['ptdtrace', 'pesex', 'pehspnon', 'peeduca', '

# Adjust citizenship categories
df_vote["prcitshp_FOREIGN BORN, NOT A CITIZEN OF"] = 0

# Add missing columns to match structure
work_cols = set(df_work.columns)
vote_cols = set(df_vote.columns)
missing_in_vote = work_cols - vote_cols
missing_in_work = vote_cols - work_cols

for col in missing_in_vote:
    df_vote[col] = 0
for col in missing_in_work:
    df_work[col] = 0
```

To ensure that the core variables in both df_vote and df_work have the same structure, we used one-hot encoding to convert categorical variables into numerical variables and added missing columns to match the structure between the two datasets.

Example:added "FOREIGN BORN, NOT A CITIZEN OF" category in df_vote, which was missing compared to df_work.

2.e.

```
scaler = MinMaxScaler()
df_work['prtage_scaled'] = scaler.fit_transform(df_work[['prtage']])
df_vote['prtage_scaled'] = scaler.transform(df_vote[['prtage']])
```

3.

```
X_work = df_work.drop(['work', 'prtage'], axis=1)  # Exclude prtage for now
y_work = df_work['work']
```

Consider four values of hyperparameter C: 0.1, 1, 5, and 10 and three kernels: linear, radial, and sigmoid. Set random seed to 26

```
kf = KFold(n_splits=5, shuffle=True, random_state=26)

# Define hyperparameters
Cs = [0.1, 1, 5, 10]
```

```
kernels = ['linear', 'rbf', 'sigmoid']

# Initialize lists to store results
cv_errors = []

for C in Cs:
    for kernel in kernels:
        model = SVC(C=C, kernel=kernel)
        errors = []
        for train_index, test_index in kf.split(X_work):
            X_train, X_test = X_work.iloc[train_index], X_work.iloc[test_index]
            y_train, y_test = y_work.iloc[train_index], y_work.iloc[test_index]
            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)
            error = 1 - accuracy_score(y_test, y_pred)
            errors.append(error)
        cv_error = np.mean(errors)
        cv_errors.append((C, kernel, cv_error))
        print(f"C={C}, Kernel={kernel}, CV Error={cv_error}")
```

```
C=0.1, Kernel=linear, CV Error=0.1398
C=0.1, Kernel=rbf, CV Error=0.15500000000000003
C=0.1, Kernel=sigmoid, CV Error=0.2098
C=1, Kernel=linear, CV Error=0.1386
C=1, Kernel=rbf, CV Error=0.14300000000000002
C=1, Kernel=sigmoid, CV Error=0.32299999999999995
C=5, Kernel=linear, CV Error=0.13840000000000002
C=5, Kernel=rbf, CV Error=0.14500000000000002
C=5, Kernel=sigmoid, CV Error=0.34219999999999995
C=10, Kernel=linear, CV Error=0.1386
C=10, Kernel=rbf, CV Error=0.14720000000000003
C=10, Kernel=sigmoid, CV Error=0.3258
```

4.

```
# Select the best model
best_model_params = min(cv_errors, key=lambda x: x[2])
print(f"Best Model: C={best_model_params[0]}, Kernel={best_model_params[1]}")
```

```
Best Model: C=5, Kernel=linear
```

5.

```
# Fit the best model on the entire dataset
best_model = SVC(C=best_model_params[0], kernel=best_model_params[1])
best_model.fit(X_work, y_work)

# Predict on the training data
y_pred = best_model.predict(X_work)

# Calculate accuracy score
accuracy = accuracy_score(y_work, y_pred)
print(f"Accuracy Score: {accuracy}")
```

```
Accuracy Score: 0.8662
```

6.

```
# Prepare df_vote for prediction
X_vote = df_vote.drop(['vote', 'prtage'], axis=1)  # Exclude prtage for now

# Ensure both datasets have the same columns
work_cols = set(X_work.columns)
vote_cols = set(X_vote.columns)
missing_in_vote = work_cols - vote_cols
missing_in_work = vote_cols - work_cols

for col in missing_in_vote:
    X_vote[col] = 0
for col in missing_in_work:
    X_work[col] = 0

# Debugging: Print column names to ensure they match
print("X_work columns:", X_work.columns)
print("X_vote columns before reindexing:", X_vote.columns)

# Reindex X_vote to match the column order of X_work
X_vote = X_vote.reindex(columns=X_work.columns)  # <-- FIX: Ensure column order matches

# Debugging: Print column names after reindexing
print("X_vote columns after reindexing:", X_vote.columns)

# Refit the model with the updated X_work
best_model = SVC(C=best_model_params[0], kernel=best_model_params[1])
best_model.fit(X_work, y_work)
```

```
# Predict imputed work schedules
imputed_work = best_model.predict(X_vote)
df_vote['imputed_work'] = imputed_work
```

```
X_work columns: Index(['ptdtrace_4 or 5 Races', 'ptdtrace_American Indian, Alaskan',
       'ptdtrace_Asian Only', 'ptdtrace_Asian-HP', 'ptdtrace_Black Only',
       'ptdtrace_Black-AI', 'ptdtrace_Hawaiian/Pacific Islander Only',
       'ptdtrace_White Only', 'ptdtrace_White-AI', 'ptdtrace_White-Asian',
       'ptdtrace_White-Black', 'ptdtrace_White-Hawaiian', 'pesex_MALE',
       'pehspnon_NON-HISPANIC', 'peeduca_11TH GRADE',
       'peeduca_12TH GRADE NO DIPLOMA', 'peeduca_1ST, 2ND, 3RD OR 4TH GRADE',
       'peeduca_5TH OR 6TH GRADE', 'peeduca_7TH OR 8TH GRADE',
       'peeduca_9TH GRADE', 'peeduca_ASSOCIATE DEGREE-ACADEMIC',
       'peeduca_ASSOCIATE DEGREE-OCCUPATIONAL/', 'peeduca_BACHELOR'S DEGREE',
       'peeduca_DOCTORATE DEGREE', 'peeduca_HIGH SCHOOL GRAD-DIPLOMA OR',
       'peeduca_LESS THAN 1ST GRADE', 'peeduca_MASTER'S DEGREE (EX: MA, MS,',
       'peeduca_PROFESSIONAL SCHOOL DEG', 'peeduca_SOME COLLEGE BUT NO DEGREE',
       'prcitshp_FOREIGN BORN, U.S. CITIZEN BY',
       'prcitshp_NATIVE, BORN ABROAD OF',
       'prcitshp_NATIVE, BORN IN PUERTO RICO OR',
       'prcitshp_NATIVE, BORN IN THE UNITED', 'ptdtrace_Black-Asian',
       'ptdtrace_W-A-HP', 'ptdtrace_W-B-AI', 'vote',
       'prcitshp_FOREIGN BORN, NOT A CITIZEN OF', 'prtage_scaled', 'work'],
      dtype='object')
X_vote columns before reindexing: Index(['ptdtrace_American Indian, Alaskan', 'ptdtrace_A
       'ptdtrace_Asian-HP', 'ptdtrace_Black Only', 'ptdtrace_Black-AI',
       'ptdtrace_Black-Asian', 'ptdtrace_Hawaiian/Pacific Islander Only',
       'ptdtrace_W-A-HP', 'ptdtrace_W-B-AI', 'ptdtrace_White Only',
       'ptdtrace_White-AI', 'ptdtrace_White-Asian', 'ptdtrace_White-Black',
       'ptdtrace_White-Hawaiian', 'pesex_MALE', 'pehspnon_NON-HISPANIC',
       'peeduca_11TH GRADE', 'peeduca_12TH GRADE NO DIPLOMA',
       'peeduca_1ST, 2ND, 3RD OR 4TH GRADE', 'peeduca_5TH OR 6TH GRADE',
       'peeduca_7TH OR 8TH GRADE', 'peeduca_9TH GRADE',
       'peeduca_ASSOCIATE DEGREE-ACADEMIC',
       'peeduca_ASSOCIATE DEGREE-OCCUPATIONAL/', 'peeduca_BACHELOR'S DEGREE',
       'peeduca_DOCTORATE DEGREE', 'peeduca_HIGH SCHOOL GRAD-DIPLOMA OR',
       'peeduca_LESS THAN 1ST GRADE', 'peeduca_MASTER'S DEGREE (EX: MA, MS,',
       'peeduca_PROFESSIONAL SCHOOL DEG', 'peeduca_SOME COLLEGE BUT NO DEGREE',
       'prcitshp_NATIVE, BORN ABROAD OF',
       'prcitshp_NATIVE, BORN IN PUERTO RICO OR',
       'prcitshp_NATIVE, BORN IN THE UNITED',
       'prcitshp_FOREIGN BORN, NOT A CITIZEN OF', 'work',
```

```
       'ptdtrace_4 or 5 Races', 'prcitshp_FOREIGN BORN, U.S. CITIZEN BY',
       'prtage_scaled', 'vote'],
      dtype='object')
X_vote columns after reindexing: Index(['ptdtrace_4 or 5 Races', 'ptdtrace_American India
       'ptdtrace_Asian Only', 'ptdtrace_Asian-HP', 'ptdtrace_Black Only',
       'ptdtrace_Black-AI', 'ptdtrace_Hawaiian/Pacific Islander Only',
       'ptdtrace_White Only', 'ptdtrace_White-AI', 'ptdtrace_White-Asian',
       'ptdtrace_White-Black', 'ptdtrace_White-Hawaiian', 'pesex_MALE',
       'pehspnon_NON-HISPANIC', 'peeduca_11TH GRADE',
       'peeduca_12TH GRADE NO DIPLOMA', 'peeduca_1ST, 2ND, 3RD OR 4TH GRADE',
       'peeduca_5TH OR 6TH GRADE', 'peeduca_7TH OR 8TH GRADE',
       'peeduca_9TH GRADE', 'peeduca_ASSOCIATE DEGREE-ACADEMIC',
       'peeduca_ASSOCIATE DEGREE-OCCUPATIONAL/', 'peeduca_BACHELOR'S DEGREE',
       'peeduca_DOCTORATE DEGREE', 'peeduca_HIGH SCHOOL GRAD-DIPLOMA OR',
       'peeduca_LESS THAN 1ST GRADE', 'peeduca_MASTER'S DEGREE (EX: MA, MS,',
       'peeduca_PROFESSIONAL SCHOOL DEG', 'peeduca_SOME COLLEGE BUT NO DEGREE',
       'prcitshp_FOREIGN BORN, U.S. CITIZEN BY',
       'prcitshp_NATIVE, BORN ABROAD OF',
       'prcitshp_NATIVE, BORN IN PUERTO RICO OR',
       'prcitshp_NATIVE, BORN IN THE UNITED', 'ptdtrace_Black-Asian',
       'ptdtrace_W-A-HP', 'ptdtrace_W-B-AI', 'vote',
       'prcitshp_FOREIGN BORN, NOT A CITIZEN OF', 'prtage_scaled', 'work'],
      dtype='object')
```

```
# Calculate summary statistics for the imputed measure
summary_stats = df_vote['imputed_work'].describe()
print(summary_stats)
```

```
count    5000.000000
mean        0.548800
std         0.497663
min         0.000000
25%         0.000000
50%         1.000000
75%         1.000000
max         1.000000
Name: imputed_work, dtype: float64
```

7.

```
import statsmodels.formula.api as smf

# Include age and age squared in the model
```

```
df_vote['prtage_squared'] = df_vote['prtage'] ** 2

result = smf.ols('vote ~ imputed_work + prtage + prtage_squared + pesex_MALE', data=df_vo
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   vote   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.567
Method:                 Least Squares   F-statistic:                     1635.
Date:                Fri, 14 Mar 2025   Prob (F-statistic):               0.00
Time:                        19:17:29   Log-Likelihood:                -1530.2
No. Observations:                5000   AIC:                             3070.
Df Residuals:                    4995   BIC:                             3103.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept              1.0963      0.046     23.970      0.000       1.007       1.186
pesex_MALE[T.True]     0.0146      0.009      1.564      0.118      -0.004       0.033
imputed_work           0.3197      0.017     18.778      0.000       0.286       0.353
prtage                -0.0205      0.002    -13.301      0.000      -0.024      -0.017
prtage_squared      7.515e-05   1.43e-05      5.245      0.000    4.71e-05       0.000
==============================================================================
Omnibus:                      173.293   Durbin-Watson:                   1.992
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              351.637
Skew:                          -0.239   Prob(JB):                     4.39e-77
Kurtosis:                       4.208   Cond. No.                     2.99e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifie
[2] The condition number is large, 2.99e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

There seems to be a strong relationship between flexibility of workschedule and voting. All other variables constant, individuals with a flexible work schedule (imputed_work = 1) are 31.97 pps more likely to vote than those without a flexible work schedule, which makes intuitive sense and is statistically significant at the 0.1% level.

Same goes for age (signifi ant at the 0.1% level). For each additional year of age, the probability of voting decreases by 2.05 pps, all else constant.

Age squared has a positive coefficient, which suggests that the negative effect of age on voting diminishes as individuals grow older, meaning there is liekly aa non-linear relationship between age and voting behavior.

Males are 1.46 pps more likely to vote than females, holding other variables constant, bit this is not statistically significant at the 5% level. This makes sense and suggests that the relationship of sex and voting behavior isn't strong.

The intercept's value is not meaningful (suggests that voting of a 0 year old female without a flesxible work schedule is 109.63%)

The model explains 56.7% of the variance.

8.

```
# Define the attenuation bias correction formula
def compute_M(a, b):
    return 1 / (1 - 2 * b) * (1 - (1 - b) * b / a - (1 - b) * b / (1 - a))

# Calculate inputs to the bias correction
a = np.mean(imputed_work)
b = best_model_params[2]  # Use the best model's CV error rate

M = compute_M(a, b)
print(f"M={M}")

# Correct the estimate
work_vote_relationship = result.params['imputed_work']
work_vote_bias_correction = work_vote_relationship / M
print(f"Corrected Estimate={work_vote_bias_correction}")
```

```
M=0.7168569256117865
Corrected Estimate=0.44603782181340046
```

9.

```
# Define the attenuation bias correction formula
def compute_M(a, b):
    return 1 / (1 - 2 * b) * (1 - (1 - b) * b / a - (1 - b) * b / (1 - a))

# Calculate inputs to the bias correction
a = np.mean(imputed_work)  # Mean of the imputed work schedule
b = best_model_params[2]   # Cross-validation error rate of the best model
```

```python
# Compute M
M = compute_M(a, b)
print(f"M = {M}")

# Correct the estimate for imputed_work
work_vote_relationship = result.params['imputed_work']  # Original coefficient from regre
work_vote_bias_correction = work_vote_relationship / M  # Corrected coefficient
print(f"Corrected Estimate = {work_vote_bias_correction}")
```

```
M = 0.7168569256117865
Corrected Estimate = 0.44603782181340046
```