

Project Title:

"The Occupancy Equation: Analyses & Study of Dwelling
Occupancy Status in the Washington State, US"

TABLE OF CONTENT

| Title | Page No |
|------------------------|---------|
| Abstract | 2 |
| Intro and Overview | 3 |
| Theoretical Background | 4 |
| Methodology | 5 |
| Computational results | 6 |
| Discussion | 12 |
| Conclusion | 15 |
| Bibliography | 16 |

ABSTRACT

This project delves into employing support vector models (SVM) for classification. Utilizing IPUMS USA which contains harmonized census and American Community Survey (ACS) data, encompassing variables related to individuals and housing, such as age, income, housing cost, and population density. Here dwelling occupancy status (owner or renter) is predicted based on various factors. Three different kernels are implemented of the Support Vector Classifier, cross validation is done to tune the parameters, and feature extraction is done to achieve the best results. The project estimates dwelling occupancy status, which helps to understand housing dynamics, guiding policy decisions, market analysis, and resource allocation. After implementing the three kernels of the Support Vector Classifier which are linear, polynomial, and radial, the radial kernel achieved the highest test accuracy of 82.64%, followed by linear kernel with test accuracy of 81.36% and then the polynomial kernel with test accuracy of 80.03%. So it appears that Support Vector Classifier with radial kernel is the best model to predict 'dwelling occupancy status (owner or renter)'

INTRO AND OVERVIEW

This report aims to investigate the application of the support vector model (SVM) in classification. Harmonized Census and American Community Survey (ACS) data is examined, focusing on housing in Washington state. Through this research, if a dwelling is occupied by owners or renters based on personal and property-related factors is estimated. IPUMS USA contains harmonized census and American Community Survey (ACS) data from 1790 to the present. The available information in the censuses and ACS varies by year, but generally includes basic demographic data (age, marital status), economic data (occupation, income), and other individual characteristics (migration, disability, veteran status). The dataset consists of about 75388 entries based on number of features, some of which are mentioned above.

THEORETICAL BACKGROUND

The machine learning model that is used for the prediction of 'dwelling occupancy status' is Support Vector Machine. The Support Vector Classifier (SVC) is a powerful supervised learning algorithm used for classification tasks. At its core, SVC aims to find the optimal hyperplane that best separates data points belonging to different classes in a high-dimensional space (James, 2023). The three commonly used kernel functions in SVC are Linear, Polynomial, and Radial Basis Function.

The linear kernel is the simplest kernel function. The linear kernel produces a decision boundary that is a hyperplane in the feature space. This hyperplane separates data points from different classes in a linear fashion (James, 2023). Linear kernels are most effective when the data can be effectively separated by a straight line, plane (in 3D), or hyperplane (in higher dimensions). The parameters used for tuning the model is the cost denoted by 'C'. C is tuned from a range of values such as 0.01, 0.1, 1, and 10. Feature selection is done to find if there is a better model with selected features.

The polynomial kernel introduces non-linearity by using polynomial functions of the original features. The decision boundary can have curves and turns, suitable for data with polynomial relationships between features and classes (James, 2023). The parameters used for tuning the model are the cost denoted by 'C' and the 'degree'. C is tuned from a range of values such as 0.01, 0.1, 1, and 10. Degree is tuned from a range of values such as 2, 3 and 4. Feature selection is done to find if there is a better model with selected features.

RBF kernel, also known as Gaussian kernel, creates a decision boundary based on the similarity of data points to a reference point (or points). It can create complex, non-linear decision boundaries, effective for capturing complex and non-linear relationships in the data (James, 2023). The parameters used for tuning the model are the cost denoted by 'C' and the 'gamma'. C is tuned from a range of values such as 0.01, 0.1, 1, and 10. Gamma is tuned from a range of values such as 0.01, 0.1, 1, and 10. Feature selection is done to find if there is a better model with selected features.

As prediction of 'dwelling occupancy status' is a classification problem, so Support Vector Classifier is implemented with different kernels to find out the best one for the data.

METHODOLOGY

In the data processing part, firstly all the rows having any missing values are dropped. Next any duplicate records in the dataframe are dropped. Then important variables required for the prediction are selected and a data frame is made out of them. Next conversion of all the values which were denoted as '9' in the Vehicles column to '0' to denote 'No vehicles available'. Next 'one-hot encoding' is done for the target variable which is 'Ownership'. Next the data is split into train and test sets with test_size of 30%. Then scaling is done for the train and test sets.

Then SVC is fitted with kernel as linear to the training data, hyperparameter tuning using cross-validation and grid search is carried out where the tuning parameter is the 'cost'. Then the best model is found and test accuracy is calculated. Also feature selection followed by hyperparameter tuning using cross-validation and grid search is done, then the best model is found and test accuracy is calculated.

Next, SVC is fitted with kernel as polynomial to the training data, carried out hyperparameter tuning using cross-validation and grid search where the tuning parameter was the 'cost' and 'degree'. The best model is found and the test accuracy is calculated. Feature selection followed by hyperparameter tuning using cross-validation and grid search is done, then found out the best model and calculated the test accuracy.

Lastly, SVC is fitted with kernel as radial to the training data, carried out hyperparameter tuning using cross-validation and grid search where the tuning parameter was the 'cost' and 'gamma'. Then the best model is found and the test accuracy is calculated. Feature selection followed by hyperparameter tuning using cross-validation and grid search is done, then the best model is found and the test accuracy is calculated.

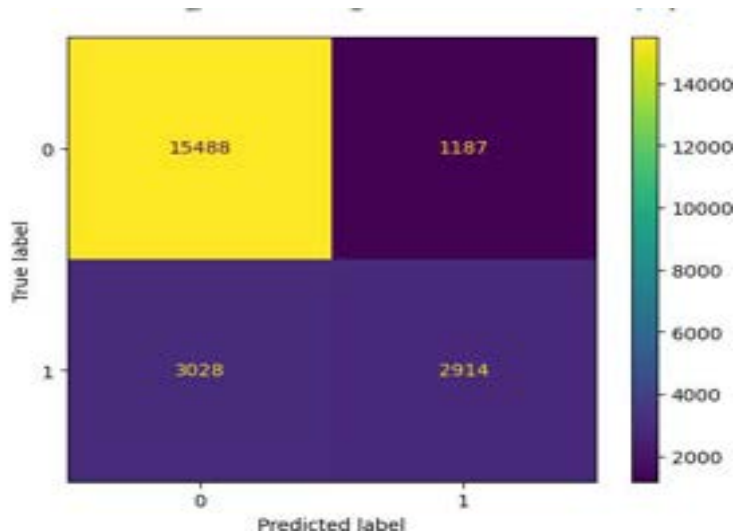
COMPUTATIONAL RESULTS

➤ SVC with kernal as linear:

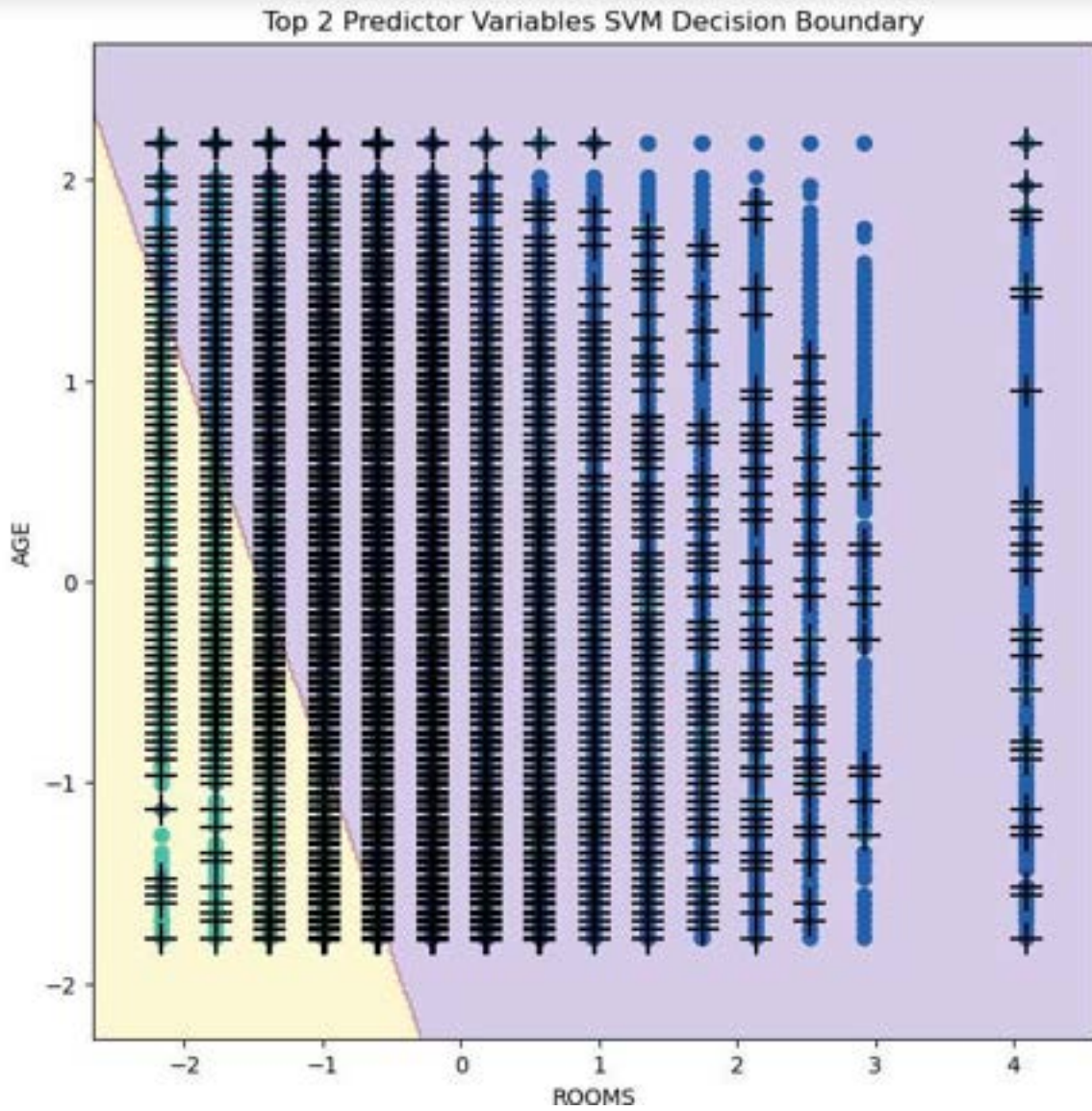
For SVC with kernal as linear, hyperparameter tuning using cross-validation and grid search is carried out. Then the best model is found and the test accuracy is calculated which is 81.36%. Feature selection followed by hyperparameter tuning using cross-validation and grid search is done, the best model is found and the test accuracy is calculated which is 80.78%. Here the model achieved without feature selection has a higher test accuracy than the model achieved with feature selection. So model achieved without feature selection appears to be a better model. The model was able to predict 'dwelling occupancy status (owner or renter)' efficiently with a test accuracy of 81.36%. Tune parameter value of cost is 0.1. The 5 most important variables found out are 'Rooms' , 'Age', 'Vehicles', 'Hhincome', and 'Nfams'.

Confusion matrix for the test set:

| Truth | 0 | 1 |
|-----------|-------|------|
| Predicted | | |
| 0 | 15488 | 3028 |
| 1 | 1187 | 2914 |



SVM Decision Boundary for top 2 predictor variables

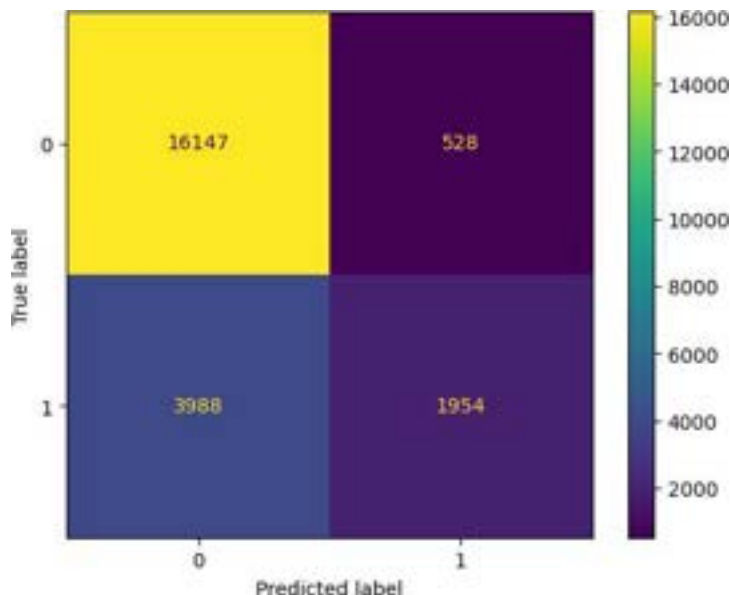


➤ SVC with kernal as polynomial

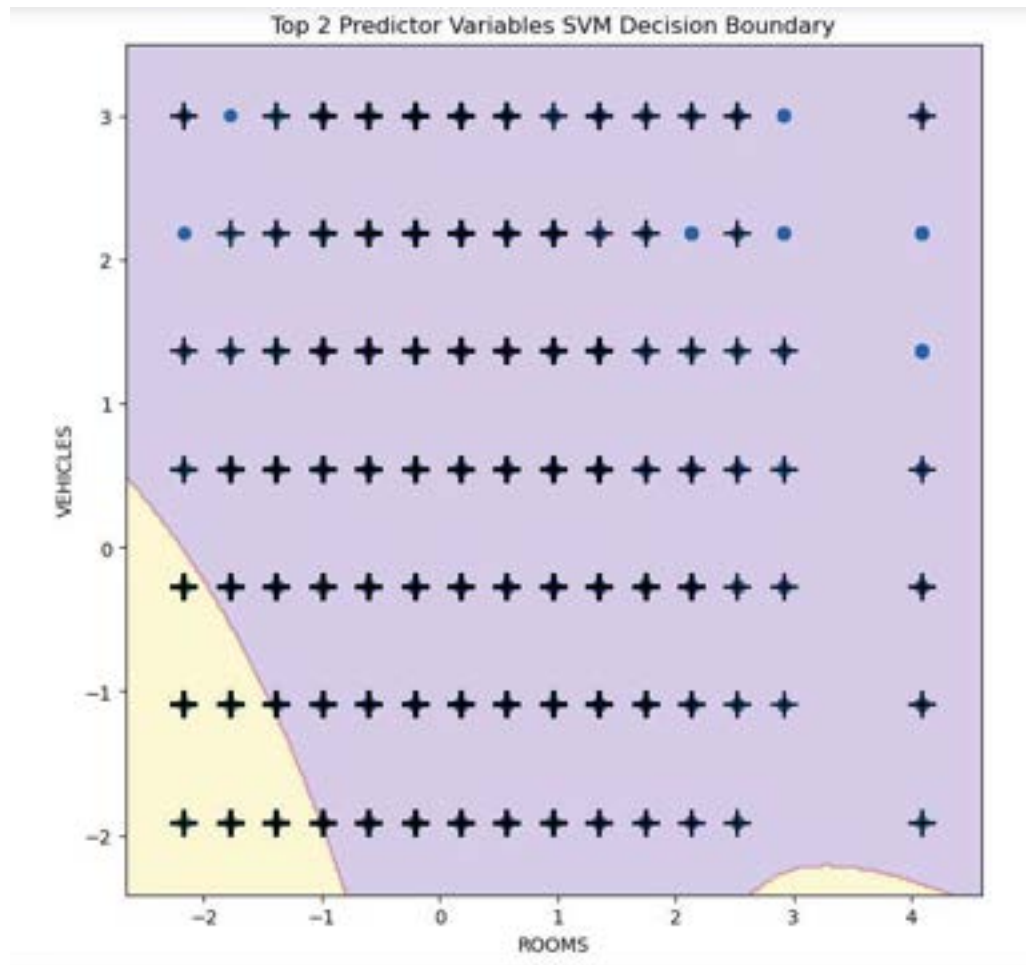
For SVC with kernal as polynomial, hyperparameter tuning using cross-validation and grid search is done. The best model is found and the test accuracy is calculated which is 80.03. Feature selection followed by hyperparameter tuning using cross-validation and grid search is done, then the best model is found and calculated the test accuracy which is 79.43. Here the model achieved without feature selection has a higher test accuracy than the model achieved with feature selection. So model achieved without feature selection appears to be a better model. The model was able to predict 'dwelling occupancy status (owner or renter)' efficiently with a test accuracy of 80.03%. Tuned value of parameters are cost is 1, degree is 3. The 5 most important variables found out are 'Rooms' , 'Vehicles', 'Age', 'Hhincome' , and 'Ncouples'

Confusion matrix for the test set:

| Truth | 0 | 1 |
|-----------|-------|------|
| Predicted | | |
| 0 | 16147 | 3988 |
| 1 | 528 | 1954 |



SVM Decision Boundary for top 2 predictor variables:

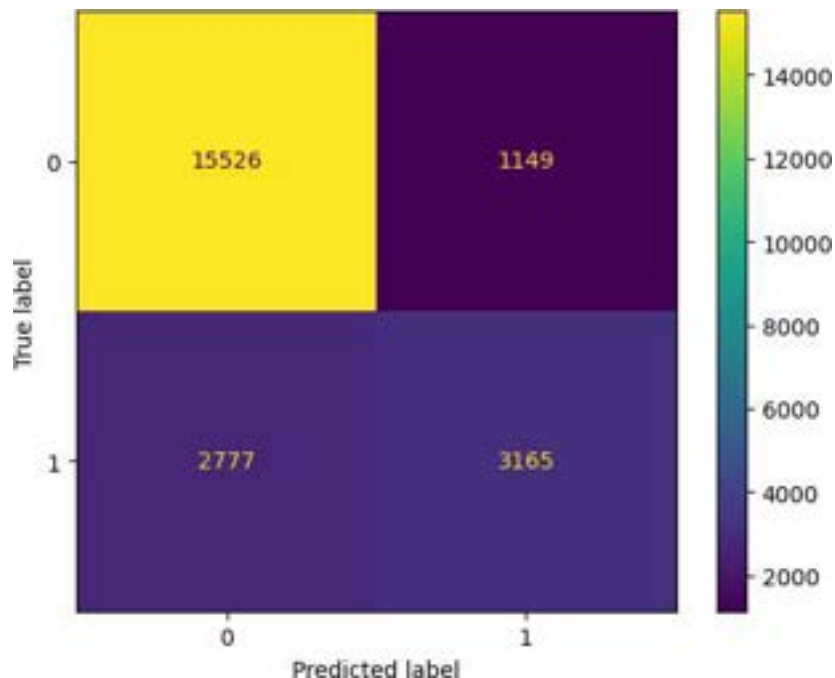


➤ SVC with kernel as radial

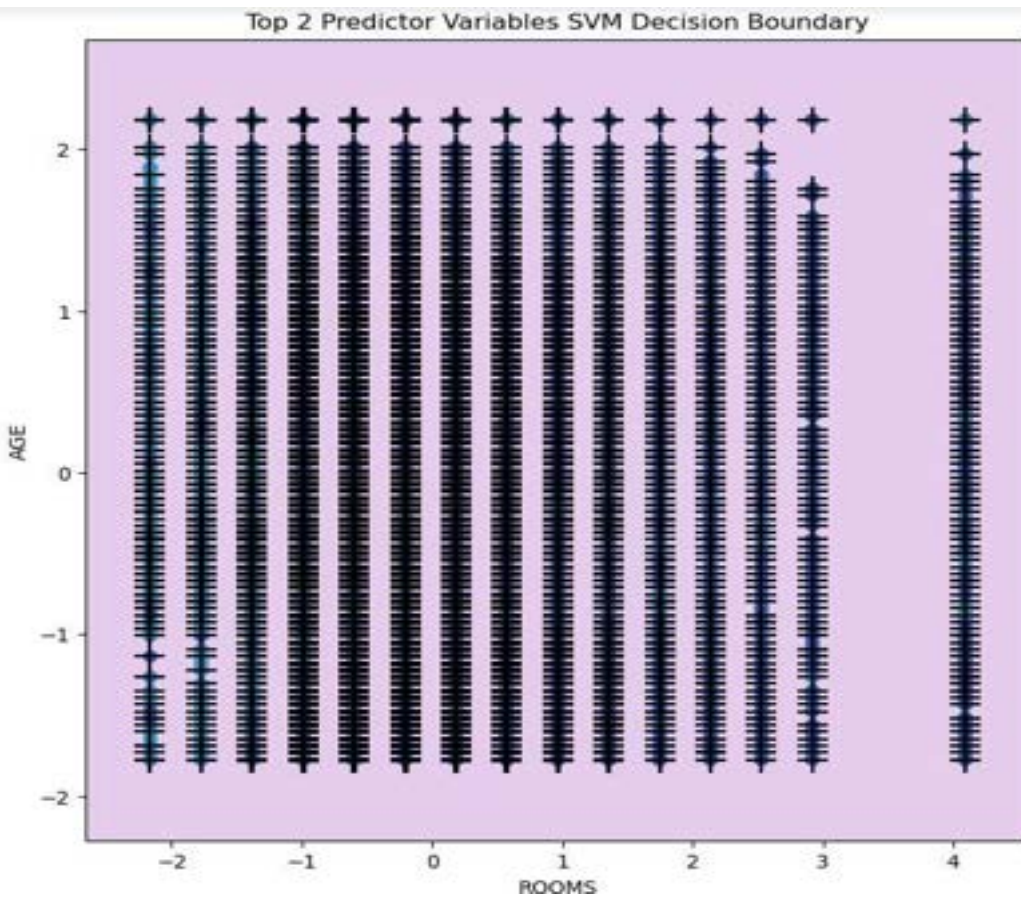
For SVC with kernel as radial, hyperparameter tuning using cross-validation and grid search is done. The best model is found and the test accuracy is calculated which is 82.64%. Feature selection followed by hyperparameter tuning using cross-validation and grid search is done, then the best model is found and the test accuracy is calculated which is 82.19%. Here the model achieved without feature selection has a higher test accuracy than the model achieved with feature selection. So model achieved without feature selection appears to be a better model. The model was able to predict 'dwelling occupancy status (owner or renter)' efficiently with a test accuracy of 82.64%. Tuned value of parameters are cost is 1, gamma is 10. The 5 most important variables found out are 'Rooms' , 'Age' , 'Vehicles' , 'Hhincome' , 'Ncouples'

Confusion matrix for the test set:

| Truth \ Predicted | 0 | 1 |
|-------------------|-------|------|
| 0 | 15526 | 2777 |
| 1 | 1149 | 3165 |



SVM Decision Boundary for top 2 predictor variables:

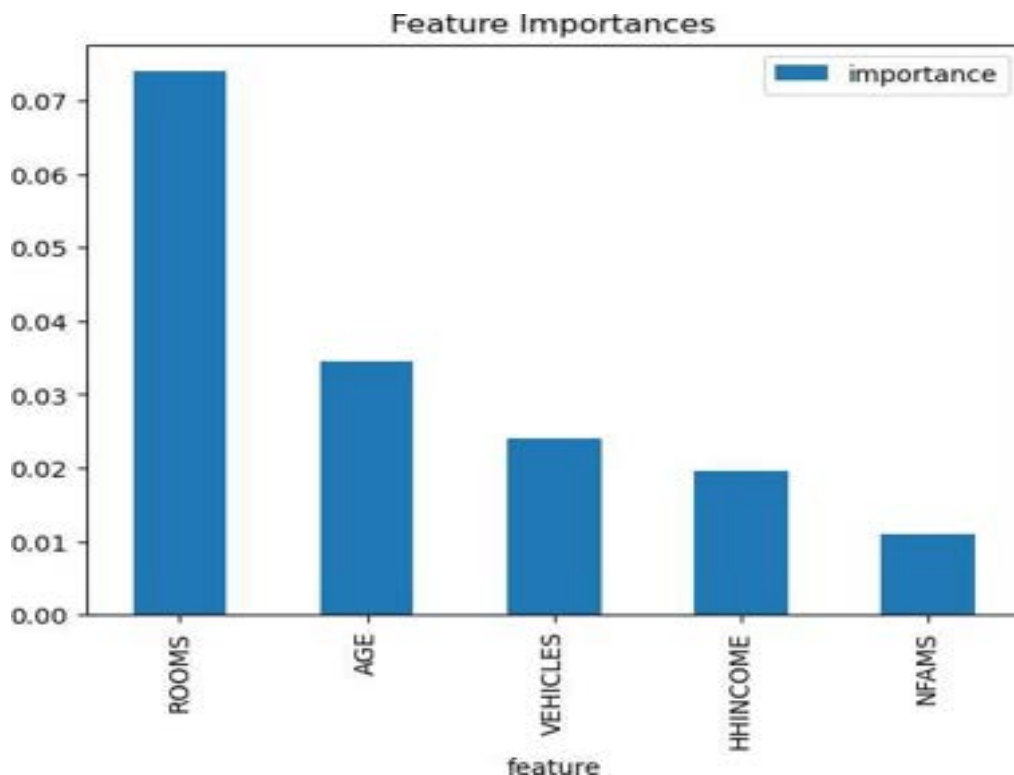


DISCUSSION

➤ SVC with kernal as linear:

The model predicted 'dwelling occupancy status (owner or renter)' efficiently with a test accuracy of 81.36%. The 5 most important variables found out are 'Rooms', 'Age', 'Vehicles', 'income' and 'nfams'. So 'Rooms' appear to be the most important variable to predict the occupancy status. It is understandable that 'number of rooms' has a clear relation with the occupancy status, as if the number of rooms are more then the dwelling could be occupied by a owner. The 'Age' is a second most important factor, and its understandable that higher the age, the more the chance of the person being the owner. 'Vehicles' is third most important factor, and it is understandable that more the number of vehicles owned, the more the chance of the person being the owner. 'Income' is fourth important factor, thus more the income is, higher is the chance of the person being a owner. Fifth important factor is 'nfams' which also clearly relates that more the number of families then more is the chance of dwelling occupied by a owner.

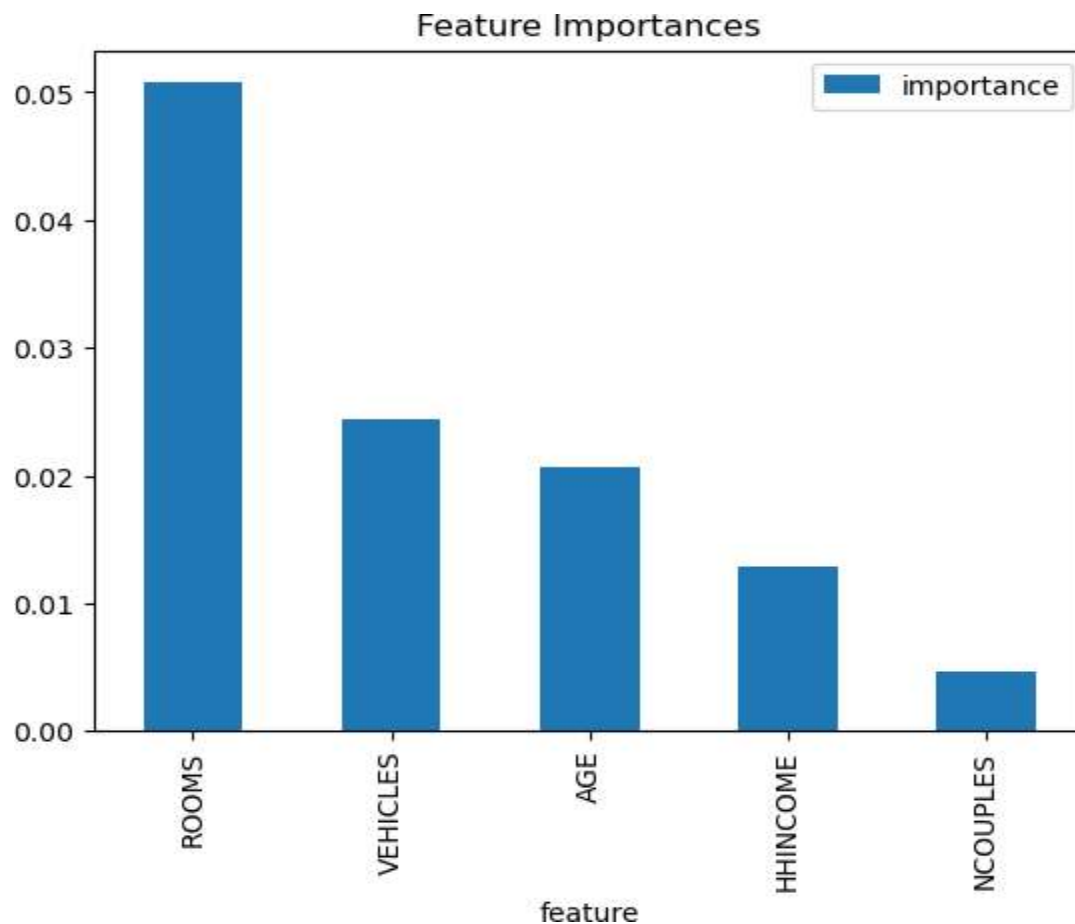
Feature importances bar graph:



➤ SVC with kernel as polynomial:

The model predicted 'dwelling occupancy status (owner or renter)' efficiently with a test accuracy of 80.03%. The 5 most important variables found out are 'Rooms' , 'Vehicles' , 'Age' , 'Hhincome' , and 'Ncouples'. So 'Rooms' appear to be the most important variable to predict the occupancy status. It is understandable that 'number of rooms' has a clear relation with the occupancy status, as if the number of rooms are more then the dwelling could be occupied by a owner. 'Vehicles' is second most important factor, and it is understandable that more the number of vehicles owned, the more the chance of the person being the owner. The 'Age' is a third most important factor, and its understandable that higher the age, the more the chance of the person being the owner. 'Income' is fourth important factor, thus more the income is, higher is the chance of the person being a owner. Fifth important factor is 'Ncouples' which also clearly relates that more the number of couples then more is the chance of dwelling occupied by a owner.

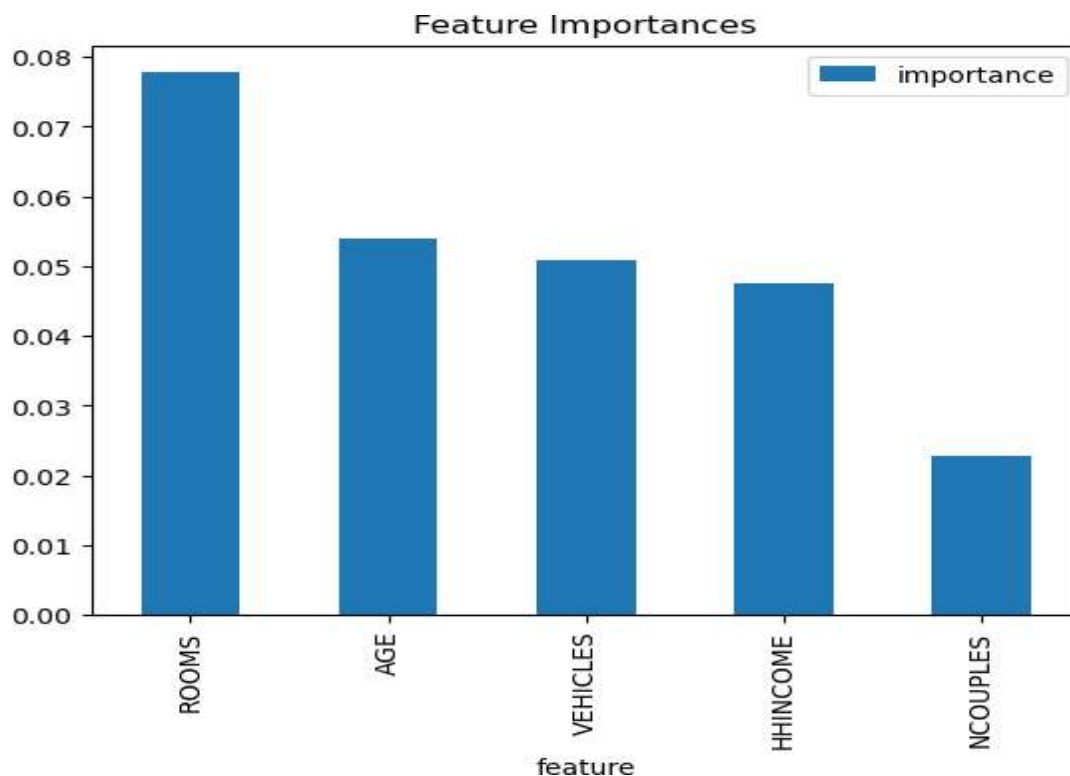
Feature importances bar graph:



➤ SVC with kernal as radial:

The model predicted 'dwelling occupancy status (owner or renter)' efficiently with a test accuracy of 82.64%. The 5 most important variables found out are 'Rooms' , 'Age' , 'Vehicles' , 'Hhincome' , 'Ncouples'. So 'Rooms' appear to be the most important variable to predict the occupancy status. It is understandable that 'number of rooms' has a clear relation with the occupancy status, as if the number of rooms are more then the dwelling could be occupied by a owner. The 'Age' is a second most important factor, and its understandable that higher the age, the more the chance of the person being the owner. 'Vehicles' is third most important factor, and it is understandable that more the number of vehicles owned, the more the chance of the person being the owner. 'Income' is fourth important factor, thus more the income is, higher is the chance of the person being a owner. Fifth important factor is 'Ncouples' which also clearly relates that more the number of couples then more is the chance of dwelling occupied by a owner.

Feature importances bar graph:



The extension that could be made to this work is I could implement Sigmoid Kernel for the support vector classifier and carry out the process of hyperparameter tuning using cross-validation and grid search.

CONCLUSION

A number of personal and property-related factors were used for investigating the application of the support vector model (SVM) to classify if a dwelling is occupied by owners or renters. A number of methods were used in the prediction of the target variables involving Support Vector Classifier which vary depending upon the kernel used such as linear, radial, and polynomial kernels. After implementing the three kernels of the Support Vector Classifier, the radial kernel achieved the highest test accuracy of 82.64%, followed by linear kernel with test accuracy of 81.36% and then the polynomial kernel with test accuracy of 80.03%. So it appears that Support Vector Classifier with radial kernel is the best model to predict 'dwelling occupancy status (owner or renter)'. The model predicted 18,691 observations correctly out of 22617. It predicted 3,926 observations incorrectly out of 22617. The 5 most important variables found out are 'Rooms', 'Age', 'Vehicles', 'Hhincome', 'Ncouples'. The number of people who own the dwelling is much higher than the number of people who rent the dwelling.

BIBLIOGRAPHY

James, W. H. (2023). *An Introduction to Statistical Learning with Applications in Python/R*.

```
In [1]: #library imports
import numpy as np
import pandas as pd
from matplotlib.pyplot import subplots, cm
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
import sklearn.model_selection as skm
from sklearn import preprocessing
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
```

```
In [2]: from ISLP import load_data, confusion_table

from sklearn.svm import SVC
from ISLP.svm import plot as plot_svm
from sklearn.metrics import RocCurveDisplay
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.inspection import permutation_importance
import matplotlib.pyplot as plt
```

```
In [3]: roc_curve = RocCurveDisplay.from_estimator # shorthand
```

```
In [4]: #loading the csv file
pd.set_option('display.max_columns', None)
df=pd.read_csv("Housing.csv")
df.columns
```

```
Out[4]: Index(['SERIAL', 'DENSITY', 'OWNERSHP', 'OWNERSHPD', 'COSTELEC', 'COSTGAS',
              'COSTWATR', 'COSTFUEL', 'HHINCOME', 'ROOMS', 'BUILTYR2', 'BEDROOMS',
              'VEHICLES', 'NFAMS', 'NCOUPLES', 'PERNUM', 'PERWT', 'AGE', 'MARST',
              'BIRTHYR', 'EDUC', 'EDUCD', 'INCTOT'],
              dtype='object')
```

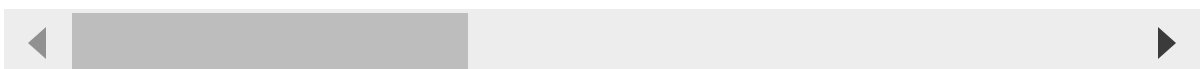
```
In [5]: df.shape
```

```
Out[5]: (75388, 23)
```

```
In [6]: df.head()
```

```
Out[6]:
```

| | SERIAL | DENSITY | OWNERSHP | OWNERSHPD | COSTELEC | COSTGAS | COSTWATR | COST |
|---|---------|---------|----------|-----------|----------|---------|----------|------|
| 0 | 1371772 | 920.0 | 1 | 13 | 9990 | 9993 | 360 | |
| 1 | 1371773 | 3640.9 | 2 | 22 | 1080 | 9993 | 1800 | |
| 2 | 1371773 | 3640.9 | 2 | 22 | 1080 | 9993 | 1800 | |
| 3 | 1371774 | 22.5 | 1 | 13 | 600 | 9993 | 9993 | |
| 4 | 1371775 | 3710.4 | 2 | 22 | 3600 | 9993 | 9997 | |



Data Preprocessing

```
In [5]: # dropping rows containing any nulls and dropping duplicate rows
df=df.dropna()
df.drop_duplicates(inplace=True)
```

```
In [6]: #creating dataframe having important variables
new_df = pd.DataFrame({
    'OWNERSHP': df['OWNERSHP'],
    'HHINCOME': df['HHINCOME'],
    'ROOMS': df['ROOMS'],
    'VEHICLES': df['VEHICLES'],
    'NFAMS': df['NFAMS'],
    'AGE': df['AGE'],
    'NCOUPLES': df['NCOUPLES'],

})
```

```
In [7]: #check for any inappropriate values
new_df['ROOMS'].unique()
```

```
Out[7]: array([ 7,  6,  5,  4,  1,  3, 10,  8,  9, 12, 17,  2, 11, 13, 14],
              dtype=int64)
```

```
In [8]: new_df['VEHICLES'].unique()
```

```
Out[8]: array([2, 3, 1, 9, 4, 5, 6], dtype=int64)
```

```
In [9]: new_df.loc[(new_df['VEHICLES']==9),['VEHICLES']]=0
```

```
In [10]: new_df['VEHICLES'].unique()
```

```
Out[10]: array([2, 3, 1, 0, 4, 5, 6], dtype=int64)
```

```
In [11]: new_df['AGE'].unique()
```

```
Out[11]: array([52, 22, 62, 50, 18, 80, 93, 61, 24, 16,  0, 45, 65, 26, 23, 74, 40,
               32,  9,  3, 39,  4, 33, 37, 41, 11,  8, 28, 36,  2, 71, 67, 58, 31,
               14, 73, 13, 10, 81, 79, 46, 12,  5, 53, 21, 51, 56, 63, 60, 47, 43,
               17, 44, 49, 87, 48, 77, 59, 27, 25,  6, 34, 35, 19, 72, 76, 29, 57,
               42, 78, 75, 66, 64, 68, 55, 70,  7, 38, 88, 54, 20, 83, 30, 84, 15,
               69, 82,  1, 85, 86, 89], dtype=int64)
```

```
In [12]: new_df['OWNERSHP'].unique()
```

```
Out[12]: array([1, 2], dtype=int64)
```

```
In [14]: #one hot encoding of target variable
latest_df=pd.get_dummies(new_df, columns=['OWNERSHP'],drop_first=True)
```

SVC with Linear kernel

Response variable: OWNERSHP_2

```
In [16]: #splitting dataset into train and test sets
(X_train,
 X_test,
 y_train,
 y_test) = skm.train_test_split(latest_df.drop(columns=['OWNERSHP_2'],axis=1),
                                latest_df['OWNERSHP_2'],
                                test_size=0.3,
                                random_state=0)
```

```
In [17]: # Scale train and test sets
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```
In [18]: print(np.mean(X_train, axis = 0).round(2))
print(np.std(X_train, axis = 0))

print(np.mean(X_test, axis = 0).round(2))
print(np.std(X_test, axis = 0).round(2))

[-0. -0. -0.  0.  0. -0.]
[1.  1.  1.  1.  1.  1.]
[ 0.01 -0.01 -0.    0.01  0.    0. ]
[1.03 0.99 0.99 1.08 1.    0.99]
```

```
In [19]: #fitting model to training data
svm_linear = SVC(C=1, kernel='linear')
svm_linear.fit(X_train, y_train)
```

```
Out[19]: SVC
SVC(C=1, kernel='linear')
```

```
In [20]: #hyperparameter tuning using cross-validation and grid search
kfold = skm.KFold(5,
                  random_state=0,
                  shuffle=True)
grid = skm.GridSearchCV(svm_linear,
                        {'C':[0.01,0.1,1,10]},
                        refit=True,
                        cv=kfold,
                        scoring='accuracy')
grid.fit(X_train, y_train)
grid.best_params_
```

```
Out[20]: {'C': 0.1}
```

```
In [21]: grid.cv_results_[('mean_test_score')]
```

```
Out[21]: array([0.81529619, 0.8153341 , 0.81529619, 0.8152583 ])
```

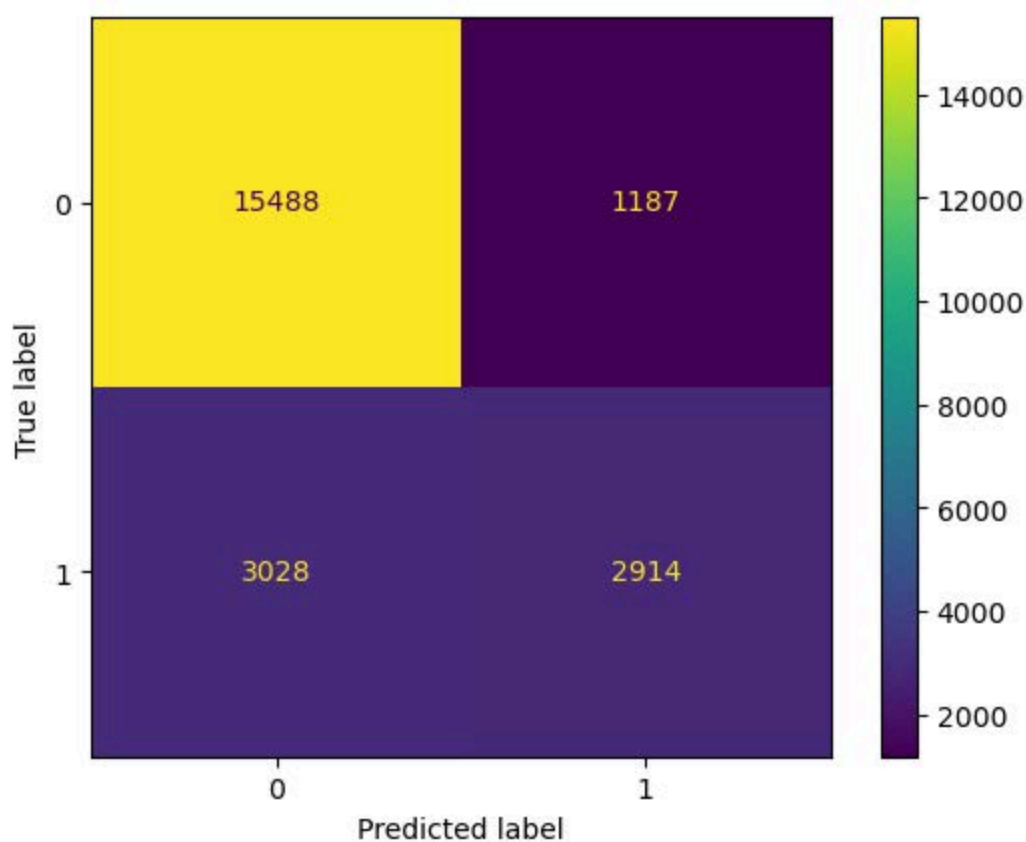
```
In [22]: # fetching the best model and finding confusion matrix for test predictions
best_ = grid.best_estimator_
y_test_hat = best_.predict(X_test)
confusion_table(y_test_hat, y_test)
```

```
Out[22]:
```

| | Truth | 0 | 1 |
|-----------|-------|------|---|
| Predicted | | | |
| 0 | 15488 | 3028 | |
| 1 | 1187 | 2914 | |

```
In [23]: ConfusionMatrixDisplay.from_predictions(y_test, y_test_hat)
```

```
Out[23]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x221d8faa650>
```



```
In [24]: #finding test accuracy
print("Accuracy: " + str(best_.score(X_test, y_test) * 100) + "%")
```

Accuracy: 81.36357607109697%

```
In [25]: #performing feature selection
selector = SelectKBest(f_classif, k=5)
selector.fit(X_train, y_train)
X_train_select = selector.transform(X_train)
X_test_select = selector.transform(X_test)
```

```
In [26]: #fitting model to selected features
svm_linear_select = SVC(C=1, kernel='linear')
svm_linear_select.fit(X_train_select, y_train)
```

```
Out[26]: SVC
SVC(C=1, kernel='linear')
```

```
In [27]: #hyperparameter tuning using cross-validation and grid search
kfold = skm.KFold(5,
                  random_state=0,
                  shuffle=True)
grid = skm.GridSearchCV(svm_linear_select,
                        {'C':[0.01,0.1,1,10]},
                        refit=True,
                        cv=kfold,
                        scoring='accuracy')
grid.fit(X_train_select, y_train)
grid.best_params_
```

```
Out[27]: {'C': 0.1}
```

```
In [28]: grid.cv_results_[('mean_test_score')]
```

```
Out[28]: array([0.8103882 , 0.81059666, 0.81053981, 0.81052086])
```

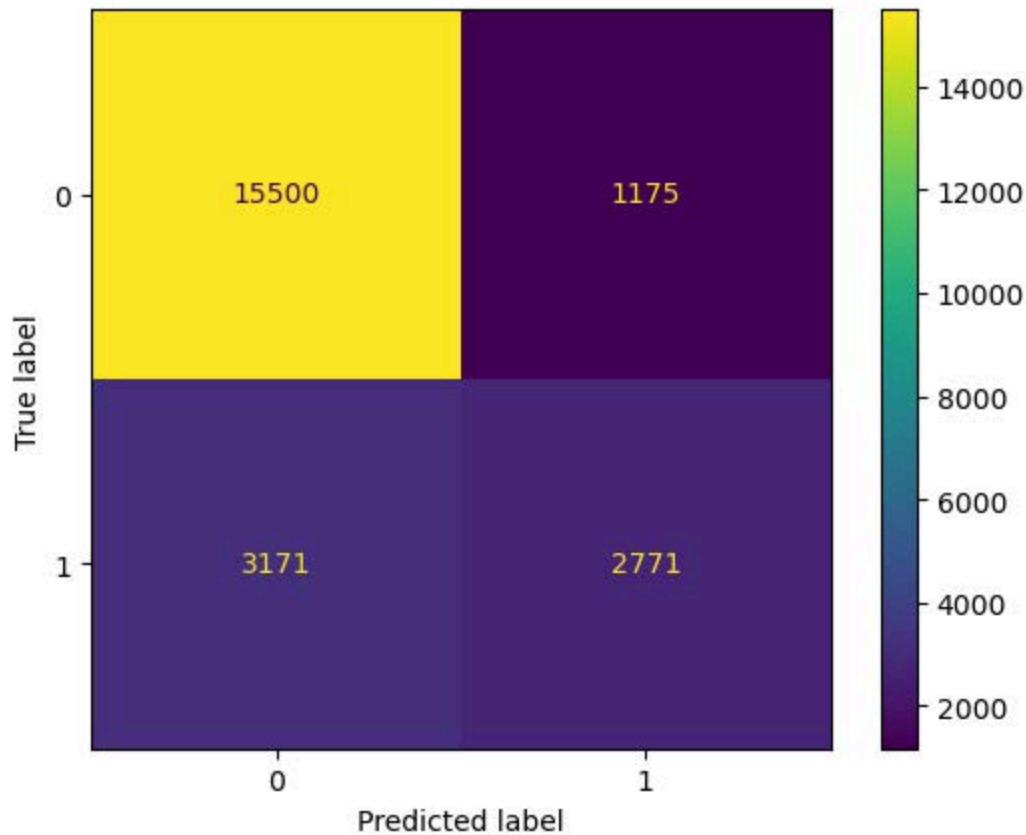
```
In [29]: # fetching the best model and finding confusion matrix for test predictions
best_select = grid.best_estimator_
y_test_hat = best_select.predict(X_test_select)
confusion_table(y_test_hat, y_test)
```

```
Out[29]:
```

| | Truth | 0 | 1 |
|-----------|-------|------|---|
| Predicted | | | |
| 0 | 15500 | 3171 | |
| 1 | 1175 | 2771 | |

```
In [30]: ConfusionMatrixDisplay.from_predictions(y_test, y_test_hat)
```

```
Out[30]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x221d8f9f0d0>
```



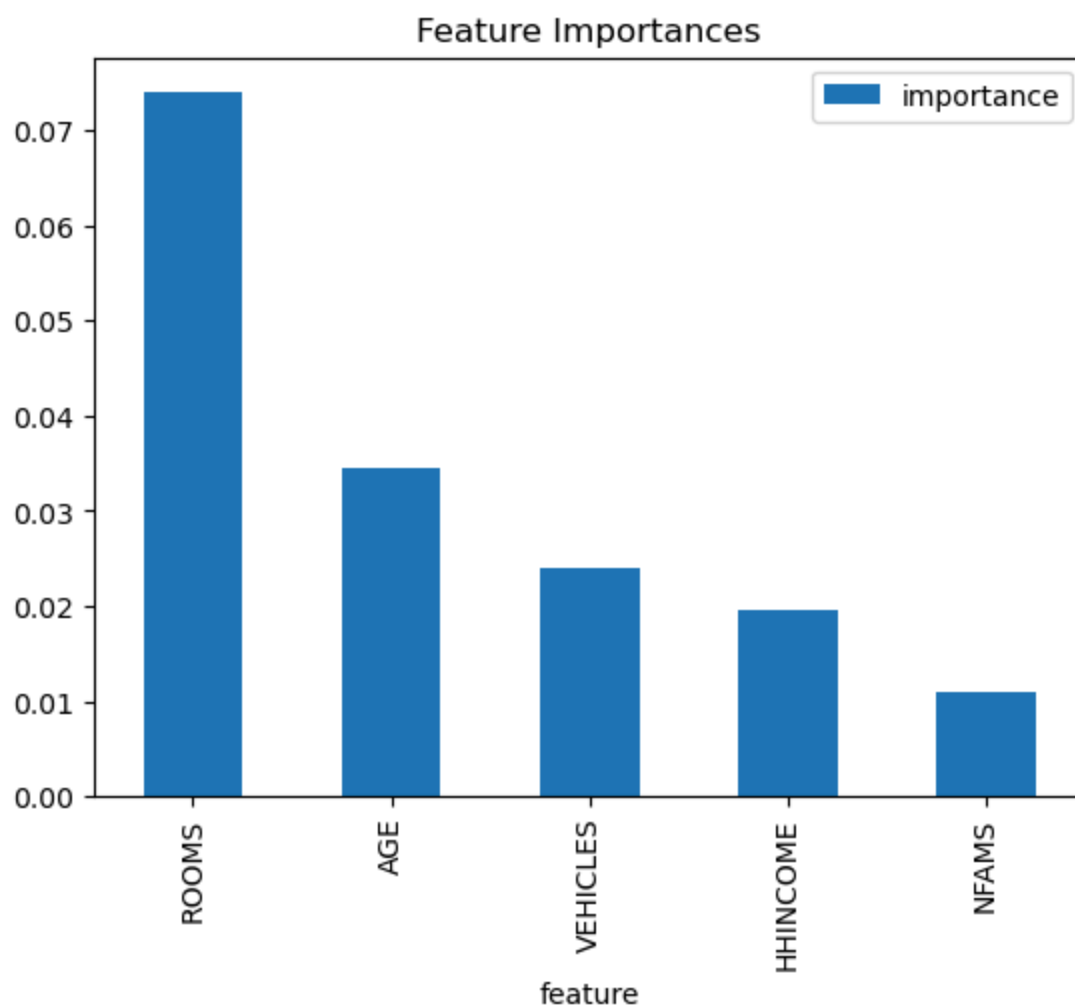
```
In [31]: #finding test accuracy
print("Accuracy: " + str(best_select.score(X_test_select, y_test) * 100) + "%")
```

Accuracy: 80.784365742583%

Here the model achieved without feature selection has a higher test accuracy than the model achieved with feature selection

```
In [32]: #creating dataframe having features and its importances
result = permutation_importance(best_, X_test, y_test, n_repeats=5, random_state=1)
importance_df = pd.DataFrame({'feature': latest_df.drop(columns=['OWNERSHP_2'],axis=1).columns, 'importance': result.importances_mean})
importance_df = importance_df.sort_values(by='importance', ascending=False).reset_index()
```

```
In [33]: #plotting bar graph for top 5 features and its importance value
importance_df.iloc[:5].plot(kind='bar', x='feature', y='importance')
plt.title('Feature Importances')
plt.show()
```

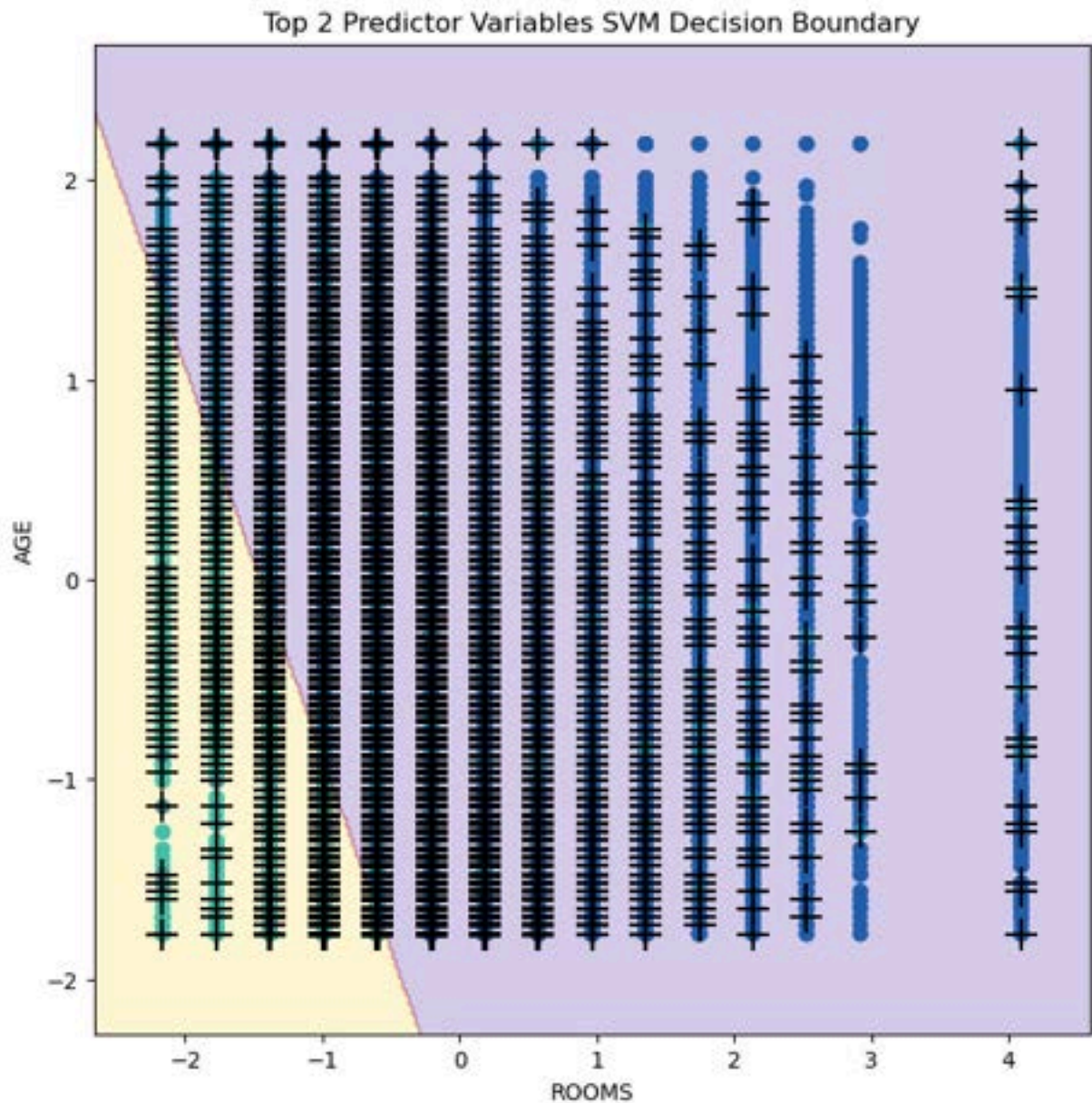


```
In [34]: # get the top 5 predictor variables
top_predictors = importance_df.head(5)['feature'].values
print("Top 5 predictor variables:", top_predictors)
```

Top 5 predictor variables: ['ROOMS' 'AGE' 'VEHICLES' 'HHINCOME' 'NFAMS']

```
In [35]: # plot of top 2 Predictor Variables SVM Decision Boundary
fig, ax = subplots(figsize=(8,8))
plot_svm(X_train,
         y_train,
         best_,
         features=(1,4),
         ax=ax)
ax.set_xlabel(top_predictors[0])
ax.set_ylabel(top_predictors[1])
ax.set_title('Top 2 Predictor Variables SVM Decision Boundary')
```

Out[35]: Text(0.5, 1.0, 'Top 2 Predictor Variables SVM Decision Boundary')



In []:

SVC with Polynomial kernel

Response variable: OWNERSHP_2

```
In [36]: #splitting dataset into train and test sets
(X_train,
 X_test,
 y_train,
 y_test) = skm.train_test_split(latest_df.drop(columns=['OWNERSHP_2'],axis=1),
                                latest_df['OWNERSHP_2'],
                                test_size=0.3,
                                random_state=0)
```

```
In [37]: # Scale train and test sets
scaler = StandardScaler().fit(X_train)
```

```
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```
In [38]: #fitting model to training data
svm_poly = SVC(C=0.01, kernel="poly", degree=2)
svm_poly.fit(X_train, y_train)
```

```
Out[38]: SVC
SVC(C=0.01, degree=2, kernel='poly')
```

```
In [39]: #hyperparameter tuning using cross-validation and grid search
kfold = skm.KFold(5,
                  random_state=0,
                  shuffle=True)
grid = skm.GridSearchCV(svm_poly,
                        {'C': [0.01, 0.1, 1],
                         'degree': [2, 3, 4]},
                        refit=True,
                        cv=kfold,
                        scoring='accuracy')
grid.fit(X_train, y_train)
grid.best_params_
```

```
Out[39]: {'C': 1, 'degree': 3}
```

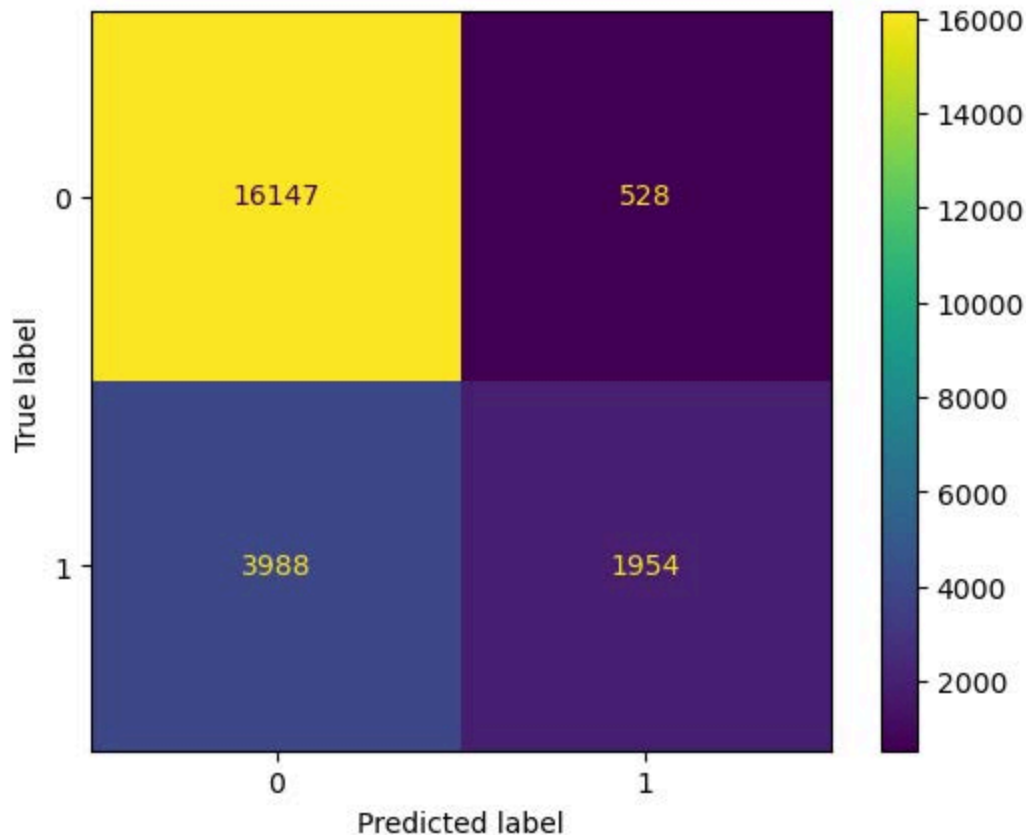
```
In [40]: # fetching the best model and finding confusion matrix for test predictions
best_ = grid.best_estimator_
y_test_hat = best_.predict(X_test)
confusion_table(y_test_hat, y_test)
```

```
Out[40]:
```

| Truth | 0 | 1 |
|-----------|-------|------|
| Predicted | | |
| 0 | 16147 | 3988 |
| 1 | 528 | 1954 |

```
In [41]: ConfusionMatrixDisplay.from_predictions(y_test, y_test_hat)
```

```
Out[41]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x221d8f659d0>
```



```
In [42]: #finding test accuracy
print("Accuracy: " + str(best_.score(X_test, y_test) * 100) + "%")
```

Accuracy: 80.03271875138171%

```
In [43]: #performing feature selection
selector = SelectKBest(f_classif, k=5)
selector.fit(X_train, y_train)
X_train_select = selector.transform(X_train)
X_test_select = selector.transform(X_test)
```

```
In [44]: #fitting model to selected features
svm_poly_select = SVC(C=0.01, kernel="poly", degree=2)
svm_poly_select.fit(X_train_select, y_train)
```

```
Out[44]: SVC
SVC(C=0.01, degree=2, kernel='poly')
```

```
In [45]: #hyperparameter tuning using cross-validation and grid search
kfold = skm.KFold(5,
                  random_state=0,
                  shuffle=True)
grid = skm.GridSearchCV(svm_poly_select,
                        {'C':[0.01,0.1,1],
                        'degree':[2,3,4]},
                        refit=True,
                        cv=kfold,
```

```

        scoring='accuracy')
grid.fit(X_train_select, y_train)
grid.best_params_

```

Out[45]: {'C': 1, 'degree': 3}

```

In [46]: # fetching the best model and finding confusion matrix for test predictions
best_select = grid.best_estimator_
y_test_hat = best_select.predict(X_test_select)
confusion_table(y_test_hat, y_test)

```

```

Out[46]:
      Truth      0      1
Predicted
      0  16228  4205
      1   447  1737

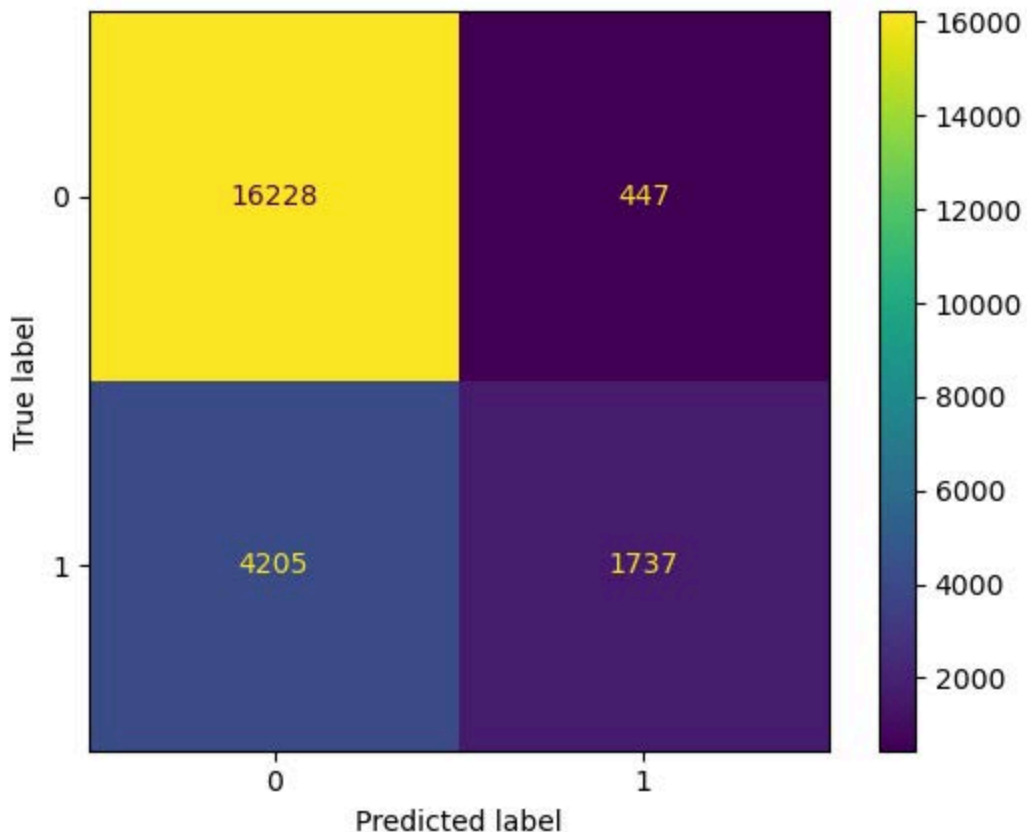
```

```

In [47]: ConfusionMatrixDisplay.from_predictions(y_test, y_test_hat)

```

Out[47]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x221dabd3cd0>



```

In [48]: #finding test accuracy
print("Accuracy: " + str(best_select.score(X_test_select, y_test) * 100) + "%")

```

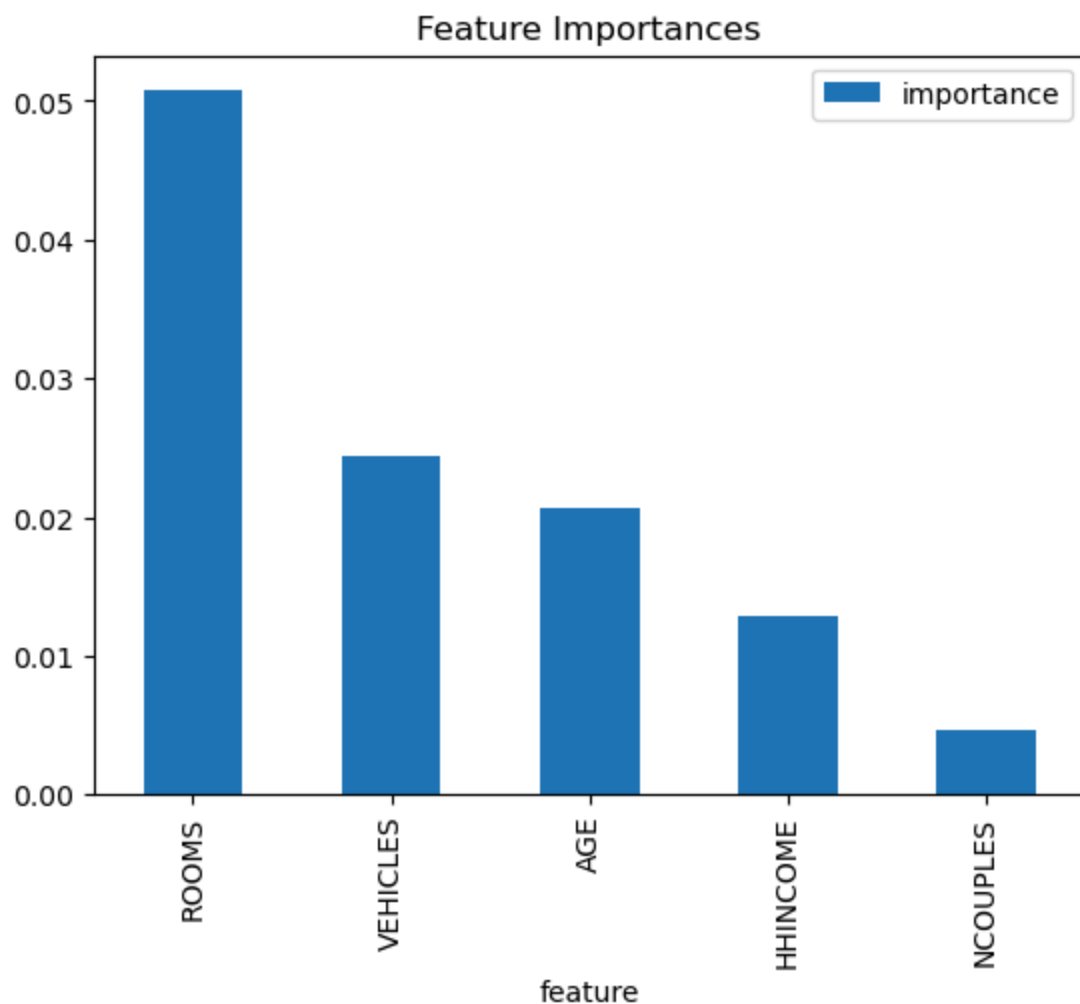
Accuracy: 79.43140115842066%

Here the model achieved without feature selection has a higher test accuracy than the model achieved with feature selection

In []:

```
In [49]: #creating dataframe having features and its importances
result = permutation_importance(best_, X_test, y_test, n_repeats=5, random_state=1)
importance_df = pd.DataFrame({'feature': latest_df.drop(columns=['OWNERSHP_2'],axis
importance_df = importance_df.sort_values(by='importance', ascending=False).reset_i
```

```
In [50]: #plotting bar graph for top 5 features and its importance value
importance_df.iloc[:5].plot(kind='bar', x='feature', y='importance')
plt.title('Feature Importances')
plt.show()
```



```
In [51]: # get the top 5 predictor variables
top_predictors = importance_df.head(5)['feature'].values
print("Top 5 predictwor variables:", top_predictors)
```

Top 5 predictwor variables: ['ROOMS' 'VEHICLES' 'AGE' 'HHINCOME' 'NCOUPLES']

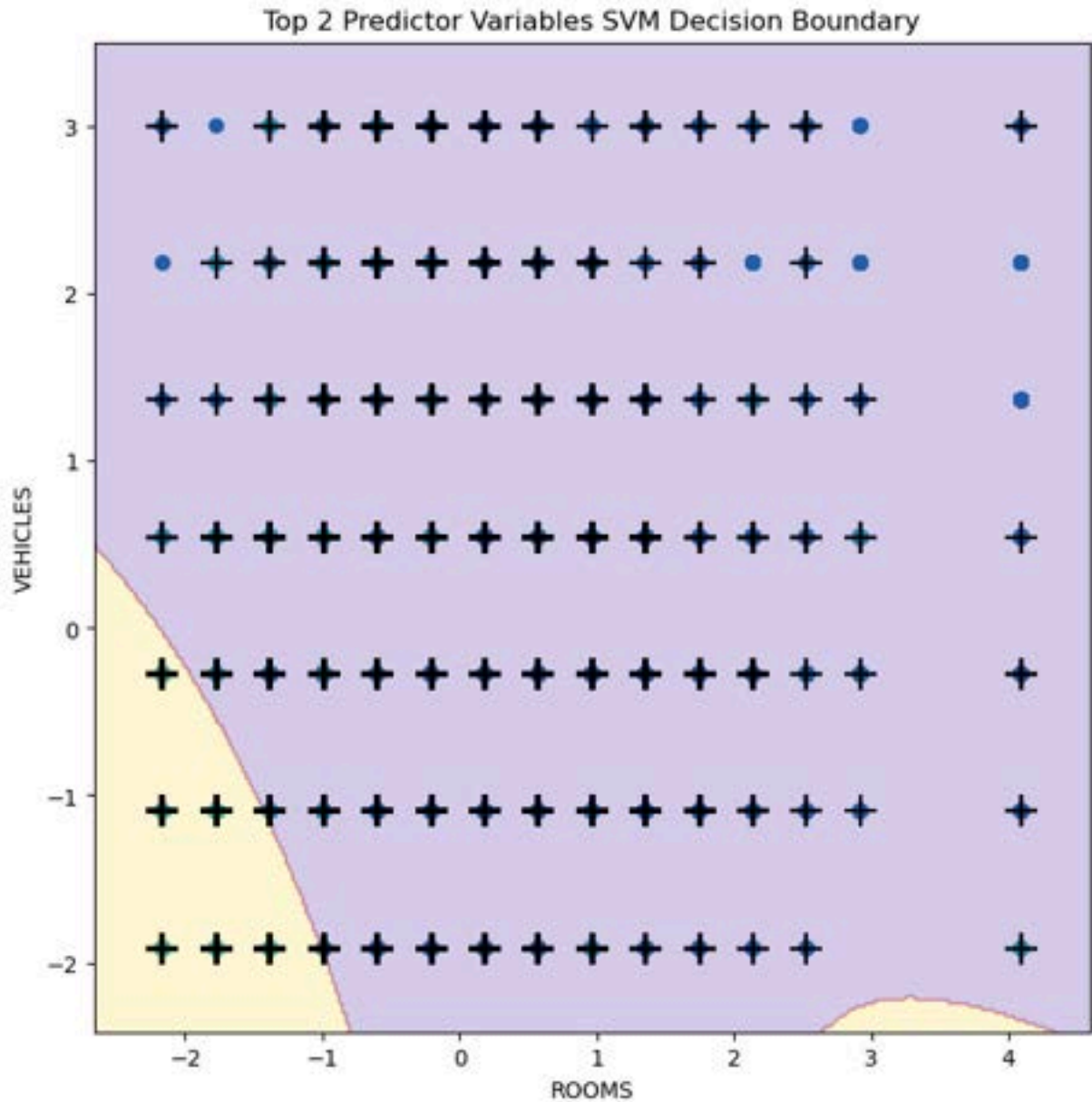
In []:

```

In [53]: # plot of top 2 Predictor Variables SVM Decision Boundary
fig, ax = subplots(figsize=(8,8))
plot_svm(X_train,
         y_train,
         best_,
         features=(1, 2),
         ax=ax)
ax.set_xlabel(top_predictors[0])
ax.set_ylabel(top_predictors[1])
ax.set_title('Top 2 Predictor Variables SVM Decision Boundary')

```

Out[53]: Text(0.5, 1.0, 'Top 2 Predictor Variables SVM Decision Boundary')



In []:

SVC with Radial kernel

Response variable: OWNERSHP_2

```
In [16]: #splitting dataset into train and test sets
(X_train,
 X_test,
 y_train,
 y_test) = skm.train_test_split(latest_df.drop(columns=['OWNERSHP_2'],axis=1),
                                latest_df['OWNERSHP_2'],
                                test_size=0.3,
                                random_state=0)
```

```
In [33]: X_test.shape
```

```
Out[33]: (22617, 6)
```

```
In [17]: # Scale train and test sets
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```
In [18]: #fitting model to training data
svm_radial = SVC(kernel="rbf", gamma=1, C=1)
svm_radial.fit(X_train, y_train)
```

```
Out[18]: SVC
SVC(C=1, gamma=1)
```

```
In [19]: #hyperparameter tuning using cross-validation and grid search
kfold = skm.KFold(5,
                  random_state=0,
                  shuffle=True)
grid = skm.GridSearchCV(svm_radial,
                        {'C':[0.01,0.1,1],
                        'gamma':[0.01, 0.1, 1, 10]},
                        refit=True,
                        cv=kfold,
                        scoring='accuracy');
grid.fit(X_train, y_train)
grid.best_params_
```

```
Out[19]: {'C': 1, 'gamma': 10}
```

```
In [20]: # fetching the best model and finding confusion matrix for test predictions
best_ = grid.best_estimator_
y_test_hat = best_.predict(X_test)
confusion_table(y_test_hat, y_test)
```

Out[20]: **Truth** **0** **1**

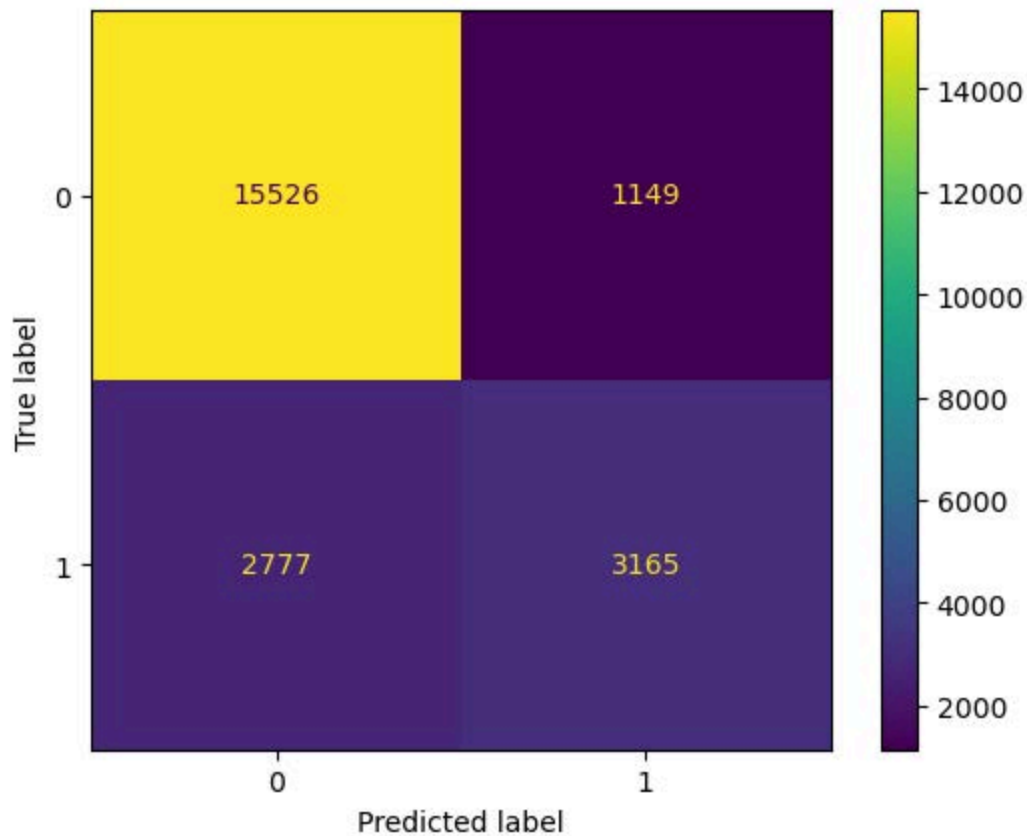
Predicted

0 15526 2777

1 1149 3165

In [21]: `ConfusionMatrixDisplay.from_predictions(y_test, y_test_hat)`

Out[21]: `<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1cf0d675090>`



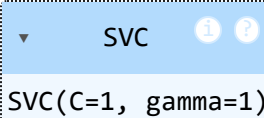
In [22]: `#finding test accuracy`
`print("Accuracy: " + str(best_.score(X_test, y_test) * 100) + "%")`

Accuracy: 82.64137595613919%

In [23]: `#performing feature selection`
`selector = SelectKBest(f_classif, k=5)`
`selector.fit(X_train, y_train)`
`X_train_select = selector.transform(X_train)`
`X_test_select = selector.transform(X_test)`

In [24]: `#fitting model to selected features`
`svm_radial_select = SVC(kernel="rbf", gamma=1, C=1)`
`svm_radial_select.fit(X_train_select, y_train)`

Out[24]:

 SVC(C=1, gamma=1)

In [25]: *#hyperparameter tuning using cross-validation and grid search*

```
kfold = skm.KFold(5,
                  random_state=0,
                  shuffle=True)
grid = skm.GridSearchCV(svm_radial_select,
                        {'C':[0.01,0.1,1],
                         'gamma':[0.01, 0.1, 1, 10]},
                        refit=True,
                        cv=kfold,
                        scoring='accuracy');
grid.fit(X_train_select, y_train)
grid.best_params_
```

Out[25]: {'C': 1, 'gamma': 10}

In [26]: *# fetching the best model and finding confusion matrix for test predictions*

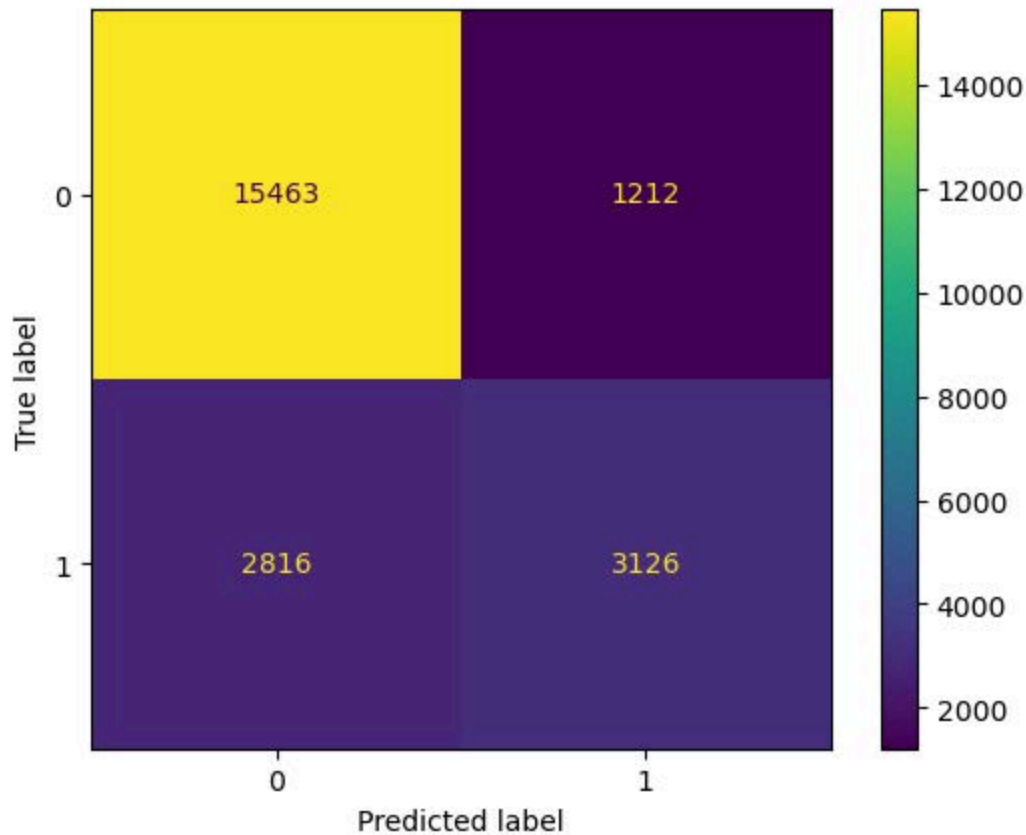
```
best_select = grid.best_estimator_
y_test_hat = best_select.predict(X_test_select)
confusion_table(y_test_hat, y_test)
```

Out[26]:

| | Truth | 0 | 1 |
|-----------|-------|------|---|
| Predicted | | | |
| 0 | 15463 | 2816 | |
| 1 | 1212 | 3126 | |

In [27]: ConfusionMatrixDisplay.from_predictions(y_test, y_test_hat)

Out[27]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1cf0db844d0>



```
In [28]: #finding test accuracy
print("Accuracy: " + str(best_select.score(X_test_select, y_test) * 100) + "%")
```

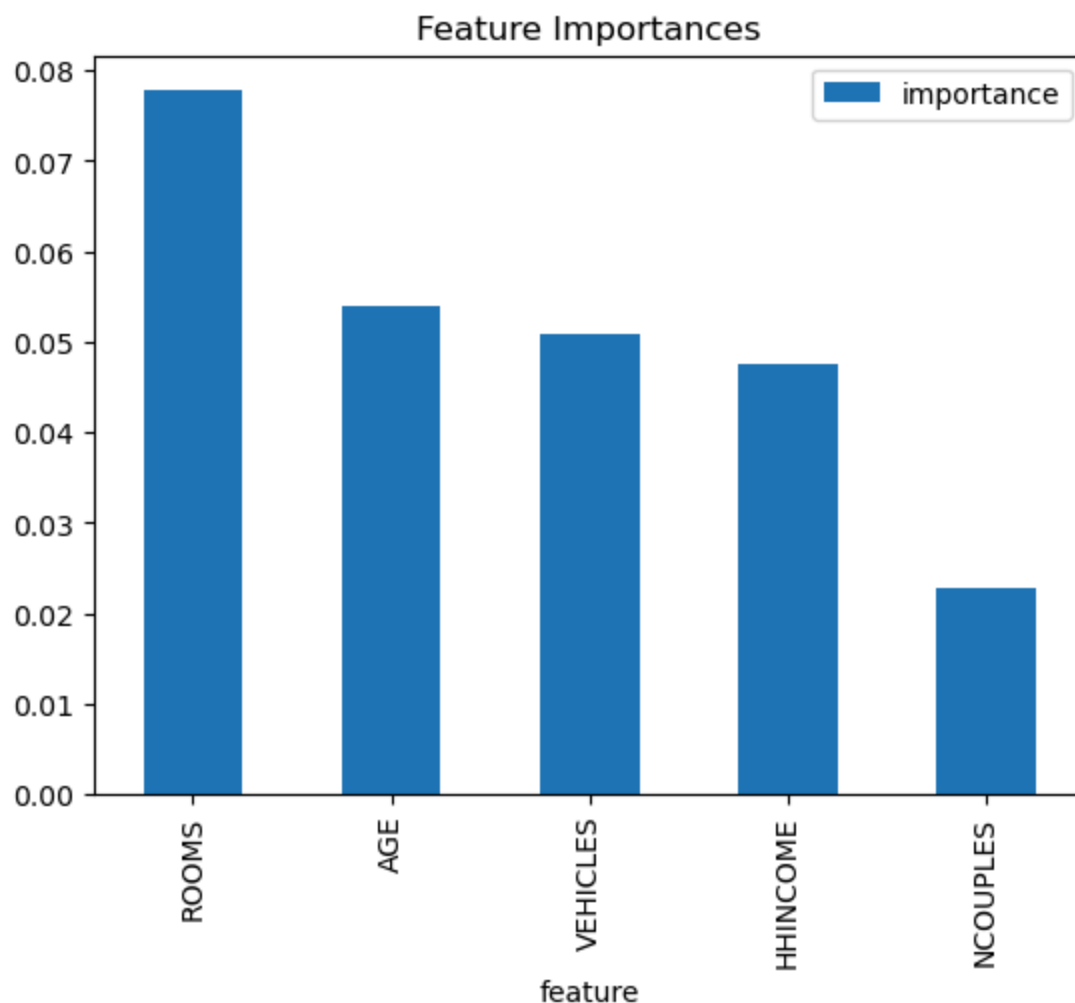
Accuracy: 82.1903877614184%

Here the model achieved without feature selection has a higher test accuracy than the model achieved with feature selection

In []:

```
In [29]: #creating dataframe having features and its importances
result = permutation_importance(best_, X_test, y_test, n_repeats=5, random_state=1)
importance_df = pd.DataFrame({'feature': latest_df.drop(columns=['OWNERSHP_2'],axis=1).columns, 'importance': result.importances_mean})
importance_df = importance_df.sort_values(by='importance', ascending=False).reset_index()
```

```
In [30]: #plotting bar graph for top 5 features and its importance value
importance_df.iloc[:5].plot(kind='bar', x='feature', y='importance')
plt.title('Feature Importances')
plt.show()
```



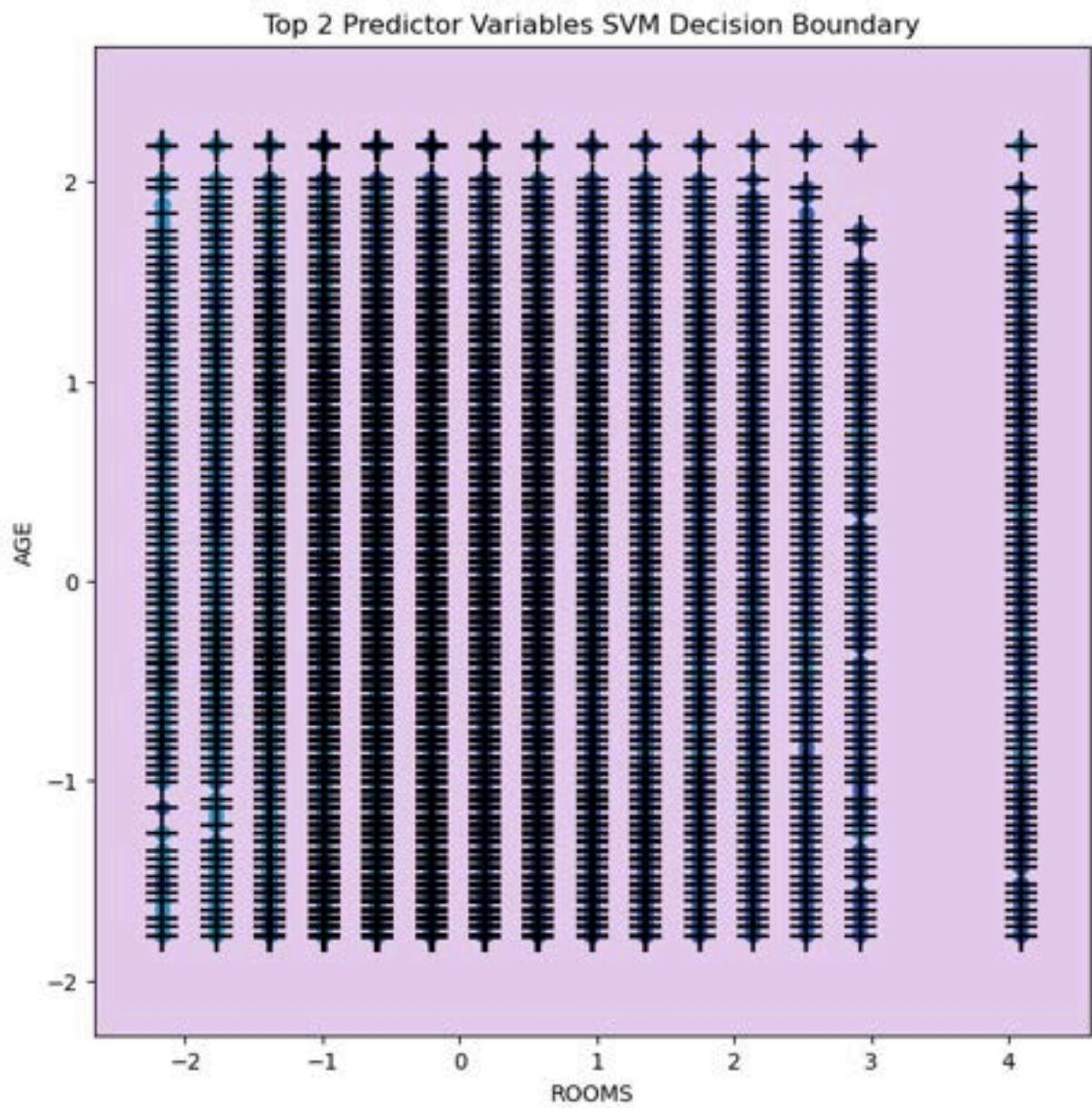
```
In [31]: # get the top 5 predictor variables
top_predictors = importance_df.head(5)['feature'].values
print("Top 5 predictor variables:", top_predictors)
```

Top 5 predictor variables: ['ROOMS' 'AGE' 'VEHICLES' 'HHINCOME' 'NCOUPLES']

```
In [ ]:
```

```
In [32]: # plot of top 2 Predictor Variables SVM Decision Boundary
fig, ax = subplots(figsize=(8,8))
plot_svm(X_train,
         y_train,
         best_,
         features=(1, 4),
         ax=ax)
ax.set_xlabel(top_predictors[0])
ax.set_ylabel(top_predictors[1])
ax.set_title('Top 2 Predictor Variables SVM Decision Boundary')
```

Out[32]: Text(0.5, 1.0, 'Top 2 Predictor Variables SVM Decision Boundary')



In []: