Support Vector Machines

Title and Abstract:

Exploring Home Ownership with Support Vector Machines: A Data Science Approach

This report takes us through the analysis of home ownership based on housing data with the help of support vector machines applied to the survey data from the Integrated Public Use Microdata Series USA. Factors related to home ownership status are learned using linear, radial and polynomial kernels. Data preprocessing, model training, parameter tuning and plots are used to build a predictive model for home ownership and learn the societal challenges around housing. It also looks at the important variables for estimating housing ownership. The models achieved accuracies of 82.80%, 83.48%, and 83.07%, with varying important variables such as number of bedrooms, number of families and age. These findings highlight the impact of support vector machines with different kernels which further help us make valuable conclusions for housing decisions.

Introduction and Overview:

The aim of this study is to use SVM models with various kernels to discuss home ownership status based on demographic and housing data. The main focus is on understanding the estimators of home ownership which is important for assessing the societal challenges that surround home ownership and for the policy makers to make appropriate decisions.

In the current society, the categorizing factors that influence home ownership status are very important for policymaking, this study aims to find the key variables using SVM models, which will explain different factors regarding home ownership decisions.

The main problems examined include which factors are important in determining whether the house is owned or rented, how much each factor is contributing in determining the status and will the policymakers get to make decisions based on this.

By using SVM models with various kernels like linear, polynomial, radial this study helps us find their effectiveness in predicting the home ownership status. Each kernel would give different effective determinants which would collectively help us build the result.

Through the analysis of SVM model results, this study wants to identify the important determinants in the home ownership decisions. These findings are expected to help policymakers learn about the factors influencing home ownership patterns which further help in making policies and housing strategies.

Theoretical Background:

“Support Vector Machines (SVM) are a class of supervised learning methods used for classification, regression and outliers’ detection. The key idea is to find the optimal

hyperplane that best separates different classes in the feature space. SVMs can handle both linear and nonlinear data by using appropriate kernel functions, such as linear, radial basis function (RBF), and polynomial kernels." [1] The data is first split into training and test set, and the training data is used to train the model and this model is used to predict the response variable and check the accuracy on the test data.

In SVM Classification, the model tries to find the hyperplane that increases the margin accordingly between the closest data points of different classes which are the support vectors. This hyperplane is found by finding the optimal parameters like cost (C) and kernel parameters such as gamma for radial kernel and degree for polynomial kernel. For multiclass classification, SVMs use one vs one and one vs all to handle multiple classes.

Parameter tuning is done to find the optimal cost, prevent overfitting and improve the model accuracy. Grid search or cross-validation techniques are usually used to find the optimal values of the parameters.

SVM classification and different kernels:

1. Linear Kernel:
   For SVM with a linear kernel, there is a straight-line decision boundary that separates the classes.
   Parameters to tune:
   Cost (C): Gives the cut-off for maximizing the margin and minimizing the error rate. Higher values of C give more complex decision boundaries which may lead to overfitting.
   The linear kernel calculates the dot product between two feature vectors for comparing the similarity between them. The decision boundary divides the space between the support vectors to maximize the margin.
2. Radial Kernel (RBF):
   For SVM with a radial kernel, a Gaussian function is used to look at the data in a higher dimension where a hyperplane is used to separate the classes.
   Parameters to tune:
   Cost (C): Gives the cut-off for maximizing the margin and minimizing the error rate. Higher values of C give more complex decision boundaries which may lead to overfitting.
   Gamma: Low values of gamma give us a large similarity radius which further leads to smooth decision boundaries.
   The radial kernel calculates the similarity between data points in feature space. When the gamma values are higher the decision boundary would be very defined which might lead to overfitting of the training data.
3. Polynomial Kernel:
   For SVM with a polynomial kernel, the similarity between two data points is calculated as the polynomial of the dot product between them.
   Parameters to tune:
   Cost (C): Gives the cut-off for maximizing the margin and minimizing the error rate. Higher values of C give more complex decision boundaries which may lead to overfitting.

Degree: Degree of the polynomial function. As the degree goes higher more complex relationships are covered but this may lead to overfitting.

The polynomial kernel gives non-linear decision boundaries by looking at the data from a high-dimensional space. As the degree increases, the model complexity increases.

Feature importance in SVM models is done by comparing the coefficients of each feature. These coefficients help us understand the contribution of each feature to the decision boundary or dividing the hyperplane. Features which have high coefficients are the most important.

Overall, Support Vector Machines perform well with large datasets and nonlinear relationships. “SVMs are widely used in applications like handwriting recognition, intrusion detection, face detection, email classification, gene classification, and in web pages. It can handle both classification and regression on linear and non-linear data.” [2]

Methodology:

Data is read from the “Housing.csv” file into a data frame “data”. The dataset was cleaned by checking each column. Considered the unique houses based on the serial number and highest age for that particular house to ensure that the same data is not being read multiple times. Modified the following columns; density, household income, marital status, and educational attainment by dividing them each into 3 or 4 categories accordingly. Dropped the irrelevant columns that won’t contribute to home ownership such as serial number, ownership of dwelling, cost of electricity, cost of gas, cost of water, cost of fuel, house value, built year and more.

The data was then divided into X and y sets where X will have all the explanatory variables after cleaning and y will have the response variables. Using the ‘train_test_split’ function the data was split into training (70%) and testing (30%).

For linear kernel, fitting with any random cost initially to find the accuracy and then do grid search to find the optimal value of C(cost) using cross-validation. A linear SVM model was trained using the optimal cost. The training and testing accuracies are calculated. Feature importance was done to identify the most influential determinants. Plotted an SVM plot on two strong predictor variables.

For radial kernel, fitting with any random cost and gamma initially to find the accuracy and then do grid search to find the optimal value of C(cost) and gamma using cross-validation. A radial SVM model was trained using the optimal cost and gamma. The training and testing accuracies are calculated. Feature importance was done to identify the most influential determinants. Plotted an SVM plot on two strong predictor variables.

For polynomial kernel, fitting with any random and degree initially to find the accuracy and then do grid search to find the optimal value of C(cost) and degree using cross-validation. A polynomial SVM model was trained using the optimal cost and degree. The training and testing accuracies are calculated. Feature importance was done to identify the most influential determinants. Plotted an SVM plot on two strong predictor variables.

The performance of each SVM model can be inferred based on training and testing accuracies. Decision boundaries were visualized for linear, radial and polynomial SVM. Support vectors and their impact on the decision boundary are presented.

Findings from the model accuracy and variable importance are analysed to help with the societal challenges that surround home ownership.

Computational Results:

1. **Linear Kernel:**
   A linear SVM model was trained using the training data. Grid Search was done to find the optimal parameters in this case C using 5-fold cross validation. The best was found to be C=0.1. Then trained the linear SVM model which gave an accuracy of 82.80% on the test data. Feature importance for the linear model was done based on coefficients. The most important features in determining the home ownership are found to be number of bedrooms and number of families. Plotted the SVM plot between them.

Feature	Importance Score
BEDROOMS	0.5933
NFAMS	0.3544

   The coefficient of "Bedrooms" (Number of Bedrooms) is -0.593302 says that as the number of bedrooms increases the probability of home ownership tend to decrease.
   The coefficient of "NFAMS" (Number of families) is 0.354441 says that as the number of families in a property decreases the probability of home ownership tend to increase.

2. **Radial Kernel:**
   A radial SVM model was trained using the training data. Grid Search was done to find the optimal parameters in this case C and gamma using 5-fold cross validation. The best was found to be C=10, gamma=0.01. Then trained the radial SVM model which gave an accuracy of 83.48% on the test data. Feature importance for the radial model was done. The most important features in determining the home ownership are found to be Age and Number of Bedrooms. Plotted the SVM plot between them.

Feature	Importance Score
AGE	0.05495
BEDROOMS	0.05412

   The importance score of "AGE" is 0.0549 says that the age of the person is highly affecting the prediction of home ownership.
   The importance score of "BEDROOMS" (Number of bedrooms) is 0.0541 says that the number of bedrooms in a house is an important predictor of home ownership.

3. **Polynomial Kernel:**
   A polynomial SVM model was trained using the training data. Grid Search was done to find the optimal parameters in this case C and degree using 5-fold cross validation.

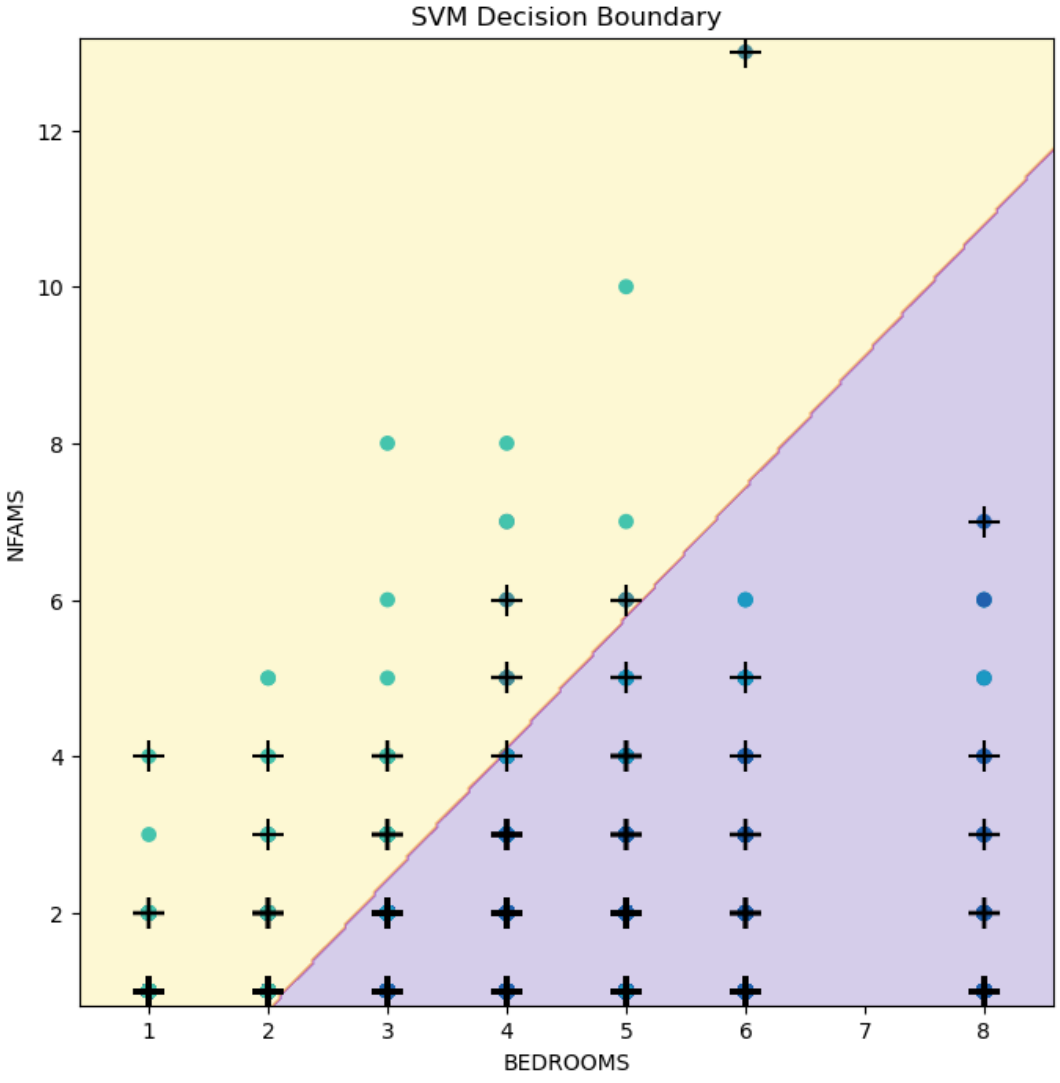
The best was found to be C=10, degree=2. Then trained the polynomial SVM model which gave an accuracy of 83.07% on the test data. Feature importance for the polynomial model was done. The most important features in determining the home ownership are found to be Number of Bedrooms and Age. Plotted the SVM plot between them.

Feature	Importance Score
BEDROOMS	0.06129
AGE	0.04713

The importance score of "BEDROOMS" (Number of bedrooms) is 0.0612 says that the number of bedrooms in a house is an important predictor of home ownership. The importance score of "AGE" is 0.0471 says that the age of the person is affecting the prediction of home ownership.

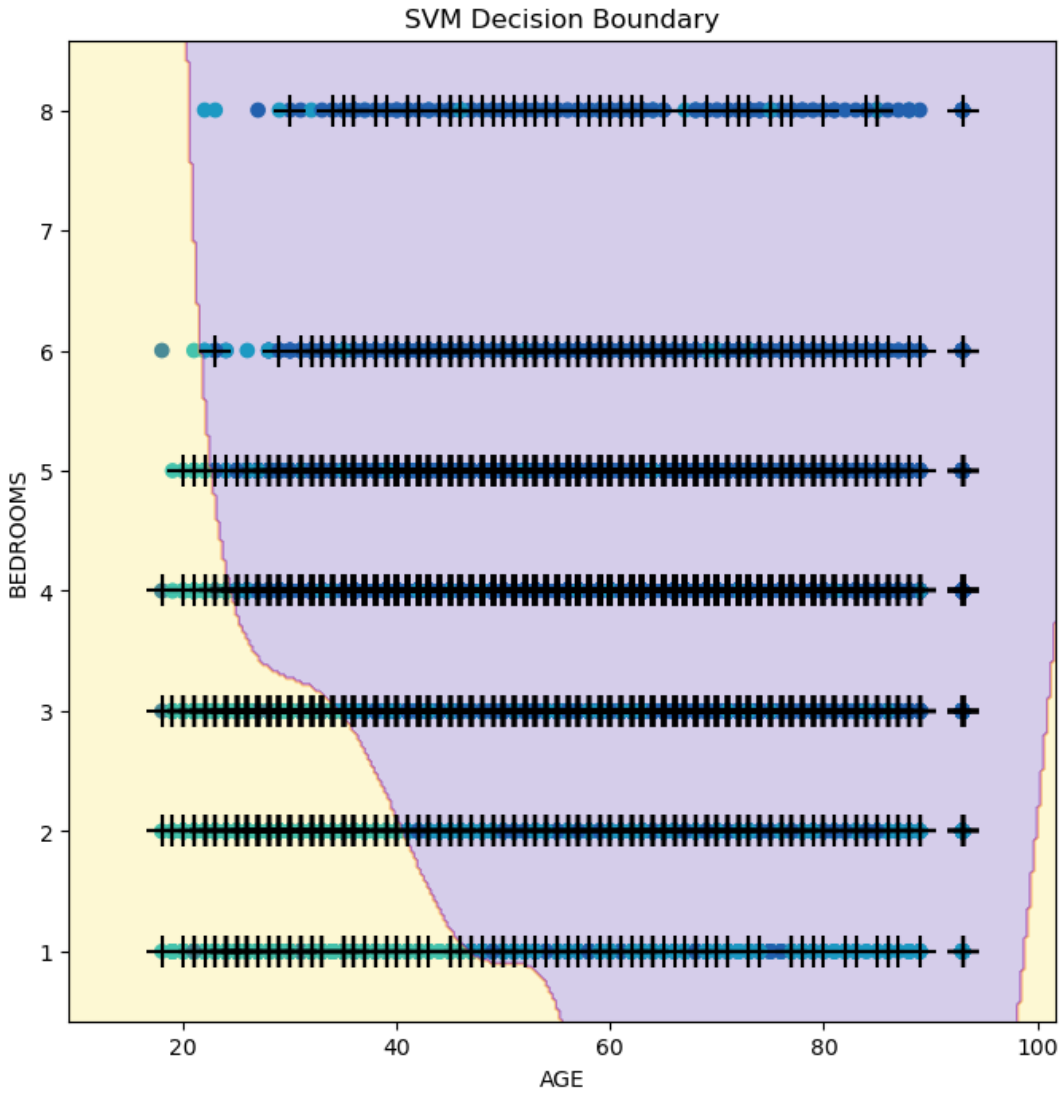
Discussion:

1. **Linear Kernel:**



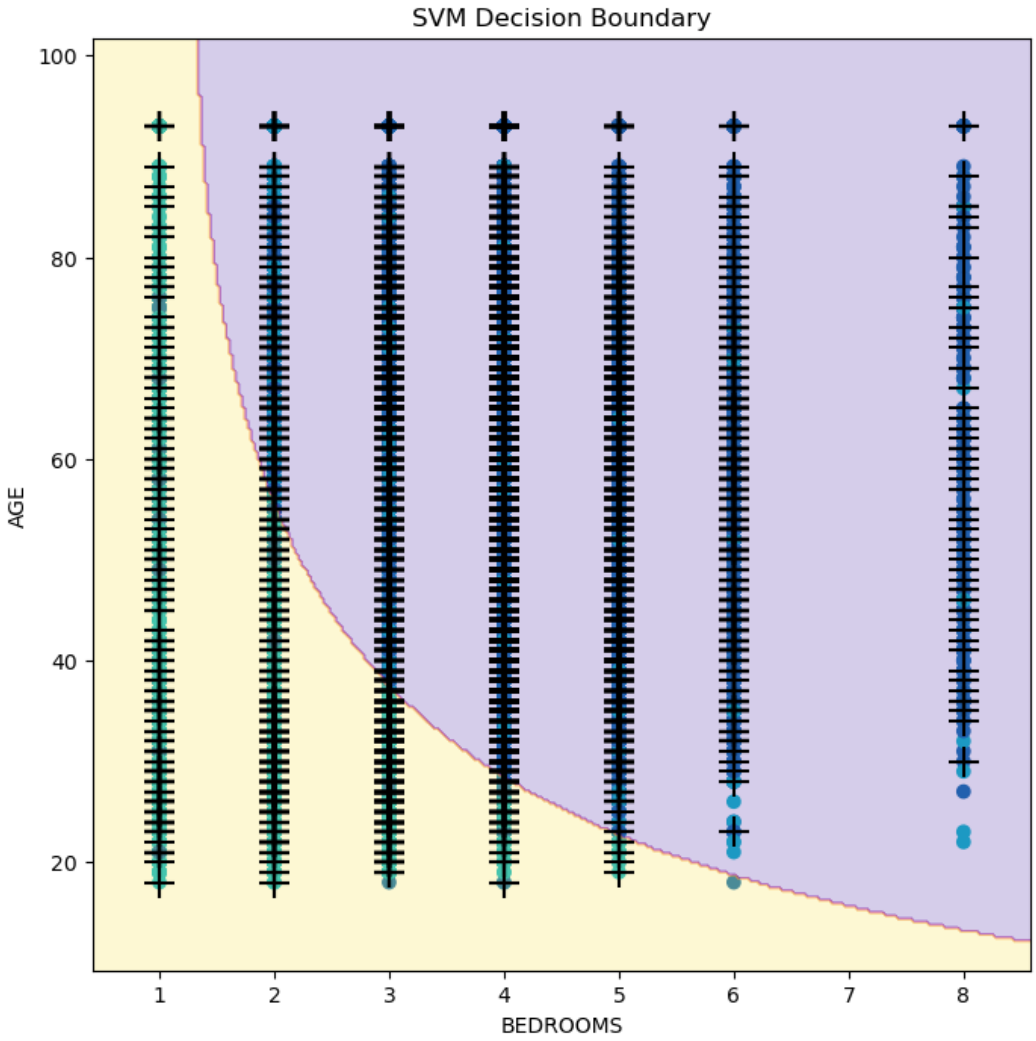
In this SVM Decision Boundary plot for linear kernel with the top 2 predictors “Bedrooms” (Number of Bedrooms) and “NFAMS” (Number of families) it can be observed that graph is divided into two classes with a line where the upper part (yellow) is the Renters Class and lower part (purple) is the Owners Class. Few points are misclassified in both the classes. But mostly the owners are misclassified as renters.

2. **Radial Kernel:**



In this SVM Decision Boundary plot for radial kernel with the top 2 predictors “AGE” (Age of the person) and “Bedrooms” (Number of Bedrooms) it can be observed that graph is divided into two classes with a curved lines where the yellow part is the Renters Class and purple part is the Owners Class. The decision boundary is drawn from a higher dimension. Few points are misclassified in both the classes.

3. **Polynomial Kernel:**



In this SVM Decision Boundary plot for polynomial kernel with the top 2 predictors “Bedrooms” (Number of Bedrooms) and “AGE” (Age of the person) it can be observed that graph is divided into two classes with a quadratic line where the yellow part is the Renters Class and purple part is the Owners Class. The decision boundary is drawn from a higher dimension where a quadratic line would divide the class. Few points are misclassified in both the classes.

Factors influencing home ownership:

For linear kernel, number of bedrooms and number of families are important factors that are influencing the home ownership. From this, it can be inferred that households with a greater number of people have difficulty in home ownership, maybe due to affordability or space requirement.

For radial kernel, age and number of bedrooms are important factors that are influencing the home ownership. From this, it can be inferred that older people may have higher probabilities of owning a home due to their savings from the past.

For polynomial kernel, number of bedrooms and age are important factors that are influencing the home ownership. From this, it can be inferred that the size of the household and age of the individual are influential.

Societal challenges surrounding home ownership:

Few societal challenges from the above findings are affordability due to a greater number of family members and wealth accumulation based on the individual's age. This further says that certain type of households has problems in attaining home ownership.

It is also observed that number of bedrooms is of high importance in all 3 kernels, which says that the household size plays an important role in home ownership.

Policy recommendations:

Based on the findings, measures for larger households and young people could be considered by the policymakers.

Policies such as providing incentives to build projects that include units with adequate bedroom sizes and implementing programs that offer support to first time or young home buyers or maintain large households could be some of the strategies that may ease the challenges.

Conclusions:

In this study, the main focus was to find the factors that influence home ownership with support vector machines (SVMs) while using 3 types of kernels (linear, radial, polynomial). SVM models were initially trained using random parameters, later grid search was done to find the best parameters and they were used. Feature importance was done to find the most important features affecting homeownership.

Linear kernel, showed number of bedrooms and number of families as the key determiners, which implied that the household size played a crucial role. Radial kernel, showed age and number of bedrooms as the key determiners, which highlight the importance of wealth accumulation in home ownership. Polynomial kernel, showed the number of bedrooms and age as the key determiners indicating importance of household size and individual age in regards of home ownership.

Altogether, the household size and demographics play an important role in home ownership. These findings can be a good start for providing appropriate directions to policymakers and stakeholders for improving the existing housing affordability, accessibility, and equity.

Bibliography/References:

1. Steven Ruggles, Sarah Flood, Matthew Sobek, Danika Brockman, Grace Cooper, Stephanie Richards, and Megan Schouweiler. IPUMS USA: Version 13.0 [dataset]. Minneapolis, MN: IPUMS, 2023. https://doi.org/10.18128/D010.V13.0
2. [1] Scikit learn, 2024, para 1: https://scikit-learn.org/stable/modules/svm.html

3. [2] Freecodecamp, https://www.freecodecamp.org/news/svm-machine-learning-tutorial-what-is-the-support-vector-machine-algorithm-explained-with-code-examples/
4. Stack Overflow, 2017. https://stackoverflow.com/questions/41592661/determining-the-most-contributing-features-for-svm-classifier-in-sklearn
5. Mingzhi Hu, Yinxin Su and Xiaofen Yu. (2022 September 7). *Homeownership and fertility intentions among migrant population in urban china.* https://www.tandfonline.com/doi/full/10.1080/02673037.2022.2108382

Appendix:

In [1]:
```
#Loading the necessary libraries
import ISLP
import numpy as np
from matplotlib.pyplot import subplots, cm
import sklearn.model_selection as skm
from ISLP import load_data, confusion_table
from sklearn.svm import SVC
from ISLP.svm import plot as plot_svm
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV
from sklearn.inspection import permutation_importance
```

In [2]:
```
#Read the data

data = pd.read_csv('Housing.csv')
print(data.head())
```

```
    SERIAL  DENSITY  OWNERSHP  OWNERSHPD  COSTELEC  COSTGAS  COSTWATR  \
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

   COSTFUEL  HHINCOME   VALUEH  ...  NFAMS  NCOUPLES  PERNUM  PERWT  AGE  \
0      9993     75000   700000  ...      1         0       1     14   52
1      9993     13600  9999999  ...      2         0       1     83   22
2      9993     13600  9999999  ...      2         0       2    106   22
3      9993      7000   800000  ...      1         0       1     33   62
4      9993     50500  9999999  ...      1         0       1    297   50

   MARST  BIRTHYR  EDUC  EDUCD  INCTOT
0      6     1969     7     71   75000
1      6     1999    10    101    5600
2      6     1999     7     71    8000
3      4     1959     6     63    7000
4      3     1971     7     71   16000

[5 rows x 24 columns]
```

In [3]:
```
data.shape
```

Out[3]:
```
(75388, 24)
```

In [4]:
```
#Considering the rows that have the highest age for that particular serial number
data = data.loc[data.groupby('SERIAL')['AGE'].idxmax()]

#Modify the DENSITY column
data['Low_DENSITY'] = (data['DENSITY'].between(0,1000, inclusive=True)).astype(int)
data['Medium_DENSITY'] = (data['DENSITY'].between(1000+1,3000, inclusive=True)).ast
data['High_DENSITY'] = (data['DENSITY'].between(3000+1,14000, inclusive=True)).asty
data.drop('DENSITY', axis=1, inplace=True)

#Modify the HHINCOME(HouseHold Income) column
data['Low_HHINCOME'] = (data['HHINCOME'].between(-7100,100000, inclusive=True)).ast
data['Medium_HHINCOME'] = (data['HHINCOME'].between(100000+1,250000, inclusive=True
data['High_HHINCOME'] = (data['HHINCOME'].between(250000+1,1700000, inclusive=True)
data.drop('HHINCOME', axis=1, inplace=True)

#Modify the MARST(Marital Status) column
```

```
data['Married'] = (data['MARST'].between(0,2, inclusive=True)).astype(int)
data['Seperated'] = (data['MARST'].between(2+1,5, inclusive=True)).astype(int)
data['Single'] = (data['MARST'].between(5+1,9, inclusive=True)).astype(int)
data.drop('MARST', axis=1, inplace=True)

#Modify the EDUC(Educational attainment) column
data['No_Education'] = (data['EDUC'] ==0).astype(int)
data['Primary_Education'] = (data['EDUC'].between(1,2, inclusive=True)).astype(int)
data['Secondary_Education'] = (data['EDUC'].between(2+1,6, inclusive=True)).astype(
data['College'] = (data['EDUC'].between(6+1,11, inclusive=True)).astype(int)
data.drop('EDUC', axis=1, inplace=True)

#Drop the columns that look like they wont contribute to Ownership
columns_to_drop = ['SERIAL', 'OWNERSHPD', 'COSTELEC', 'COSTGAS', 'COSTWATR', 'COSTF
                   'VALUEH', 'BUILTYR2', 'PERNUM', 'PERWT', 'BIRTHYR', 'EDUCD', 'IN
data = data.drop(columns_to_drop, axis=1)
```

```
C:\Users\neela\AppData\Local\Temp\ipykernel_6492\2592000663.py:5: FutureWarning: B
oolean inputs to the `inclusive` argument are deprecated in favour of `both` or `n
either`.
  data['Low_DENSITY'] = (data['DENSITY'].between(0,1000, inclusive=True)).astype(i
nt)
C:\Users\neela\AppData\Local\Temp\ipykernel_6492\2592000663.py:6: FutureWarning: B
oolean inputs to the `inclusive` argument are deprecated in favour of `both` or `n
either`.
  data['Medium_DENSITY'] = (data['DENSITY'].between(1000+1,3000, inclusive=True)).
astype(int)
C:\Users\neela\AppData\Local\Temp\ipykernel_6492\2592000663.py:7: FutureWarning: B
oolean inputs to the `inclusive` argument are deprecated in favour of `both` or `n
either`.
  data['High_DENSITY'] = (data['DENSITY'].between(3000+1,14000, inclusive=True)).a
stype(int)
C:\Users\neela\AppData\Local\Temp\ipykernel_6492\2592000663.py:11: FutureWarning:
Boolean inputs to the `inclusive` argument are deprecated in favour of `both` or `
neither`.
  data['Low_HHINCOME'] = (data['HHINCOME'].between(-7100,100000, inclusive=True)).
astype(int)
C:\Users\neela\AppData\Local\Temp\ipykernel_6492\2592000663.py:12: FutureWarning:
Boolean inputs to the `inclusive` argument are deprecated in favour of `both` or `
neither`.
  data['Medium_HHINCOME'] = (data['HHINCOME'].between(100000+1,250000, inclusive=T
rue)).astype(int)
C:\Users\neela\AppData\Local\Temp\ipykernel_6492\2592000663.py:13: FutureWarning:
Boolean inputs to the `inclusive` argument are deprecated in favour of `both` or `
neither`.
  data['High_HHINCOME'] = (data['HHINCOME'].between(250000+1,1700000, inclusive=Tr
ue)).astype(int)
C:\Users\neela\AppData\Local\Temp\ipykernel_6492\2592000663.py:17: FutureWarning:
Boolean inputs to the `inclusive` argument are deprecated in favour of `both` or `
neither`.
  data['Married'] = (data['MARST'].between(0,2, inclusive=True)).astype(int)
C:\Users\neela\AppData\Local\Temp\ipykernel_6492\2592000663.py:18: FutureWarning:
Boolean inputs to the `inclusive` argument are deprecated in favour of `both` or `
neither`.
  data['Seperated'] = (data['MARST'].between(2+1,5, inclusive=True)).astype(int)
C:\Users\neela\AppData\Local\Temp\ipykernel_6492\2592000663.py:19: FutureWarning:
Boolean inputs to the `inclusive` argument are deprecated in favour of `both` or `
neither`.
  data['Single'] = (data['MARST'].between(5+1,9, inclusive=True)).astype(int)
C:\Users\neela\AppData\Local\Temp\ipykernel_6492\2592000663.py:24: FutureWarning:
Boolean inputs to the `inclusive` argument are deprecated in favour of `both` or `
neither`.
  data['Primary_Education'] = (data['EDUC'].between(1,2, inclusive=True)).astype(i
nt)
C:\Users\neela\AppData\Local\Temp\ipykernel_6492\2592000663.py:25: FutureWarning:
Boolean inputs to the `inclusive` argument are deprecated in favour of `both` or `
neither`.
  data['Secondary_Education'] = (data['EDUC'].between(2+1,6, inclusive=True)).asty
pe(int)
C:\Users\neela\AppData\Local\Temp\ipykernel_6492\2592000663.py:26: FutureWarning:
Boolean inputs to the `inclusive` argument are deprecated in favour of `both` or `
neither`.
  data['College'] = (data['EDUC'].between(6+1,11, inclusive=True)).astype(int)
```

In [4]:
```
print(data)
```

```
       OWNERSHP  ROOMS  BEDROOMS  VEHICLES  NFAMS  NCOUPLES  AGE  Low_DENSITY  \
0             1      7         4         2      1         0   52            1
1             2      6         4         2      2         0   22            0
3             1      5         4         2      1         0   62            1
4             2      4         3         2      1         0   50            0
7             1      5         4         2      1         1   93            1
...         ...    ...       ...       ...    ...       ...  ...          ...
75373         1      7         4         3      1         0   51            1
75375         1      6         4         1      1         1   65            1
75378         1      8         5         3      1         1   70            0
75382         1      6         4         2      1         1   70            0
75386         2      8         5         2      1         2   64            0

       Medium_DENSITY  High_DENSITY  Low_HHINCOME  Medium_HHINCOME  \
0                   0             0             1                0
1                   0             1             1                0
3                   0             0             1                0
4                   0             1             1                0
7                   0             0             0                1
...               ...           ...           ...              ...
75373               0             0             0                1
75375               0             0             0                1
75378               1             0             0                1
75382               1             0             1                0
75386               1             0             1                0

       High_HHINCOME  Married  Seperated  Single  No_Education  \
0                  0        0          0       1             0
1                  0        0          0       1             0
3                  0        0          1       0             0
4                  0        0          1       0             0
7                  0        1          0       0             0
...              ...      ...        ...     ...           ...
75373              0        0          1       0             0
75375              0        1          0       0             0
75378              0        1          0       0             0
75382              0        1          0       0             0
75386              0        1          0       0             0

       Primary_Education  Secondary_Education  College
0                      0                    0        1
1                      0                    0        1
3                      0                    1        0
4                      0                    0        1
7                      0                    0        1
...                  ...                  ...      ...
75373                  0                    1        0
75375                  0                    0        1
75378                  0                    0        1
75382                  0                    1        0
75386                  0                    1        0

[30802 rows x 20 columns]
```

In [5]:

```
X= data.drop('OWNERSHP', axis=1)
y = data['OWNERSHP']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta

print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (21561, 19)
y_train shape: (21561,)
X_test shape: (9241, 19)
y_test shape: (9241,)
```

Linear

In [21]:
```
svm_linear = SVC(C=1, kernel='linear')
svm_linear.fit(X, y)
```

Out[21]:
```
SVC
SVC(C=1, kernel='linear')
```

In [22]:
```
svm_linear.support_vectors_.shape[0]
```

Out[22]:
```
13085
```

In [23]:
```
accuracy_linear_train = svm_linear.score(X_train, y_train)
accuracy_linear_train
```

Out[23]:
```
0.8259820973053198
```

In [24]:
```
accuracy_linear_test = svm_linear.score(X_test, y_test)
accuracy_linear_test
```

Out[24]:
```
0.8294556866140028
```

In [11]:
```
param_grid = {'C': [0.1, 1, 10]}

svm_linear = SVC(kernel='linear')
grid_search = GridSearchCV(estimator=svm_linear, param_grid=param_grid, cv=5, scori
grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
print("Best Parameters:", best_params)
best_score = grid_search.best_score_
print("Best Mean Cross-validated Score:", best_score)
```

```
Best Parameters: {'C': 0.1}
Best Mean Cross-validated Score: 0.826121119779826
```

In [9]:
```
svm_linear = SVC(C=0.1, kernel='linear')
svm_linear.fit(X_train, y_train)
```

Out[9]:
```
SVC
SVC(C=0.1, kernel='linear')
```

In [18]:
```
train_accuracy_linear = svm_linear.score(X_train, y_train)
train_accuracy_linear
```

Out[18]:
```
0.8262603775335096
```

In [19]:
```
test_accuracy_linear = svm_linear.score(X_test, y_test)
test_accuracy_linear
```

Out[19]:
```
0.8280489124553619
```

Feature Importance for Linear model

In [20]:
```
svm_linear = SVC(C=0.1, kernel='linear')
svm_linear.fit(X_train, y_train)

coefficients = svm_linear.coef_
feature_names = X_train.columns

coefficients_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficier
coefficients_df['Absolute_Coefficient'] = coefficients_df['Coefficient'].abs()
coefficients_df = coefficients_df.sort_values(by='Absolute_Coefficient', ascending=

print(coefficients_df)
```

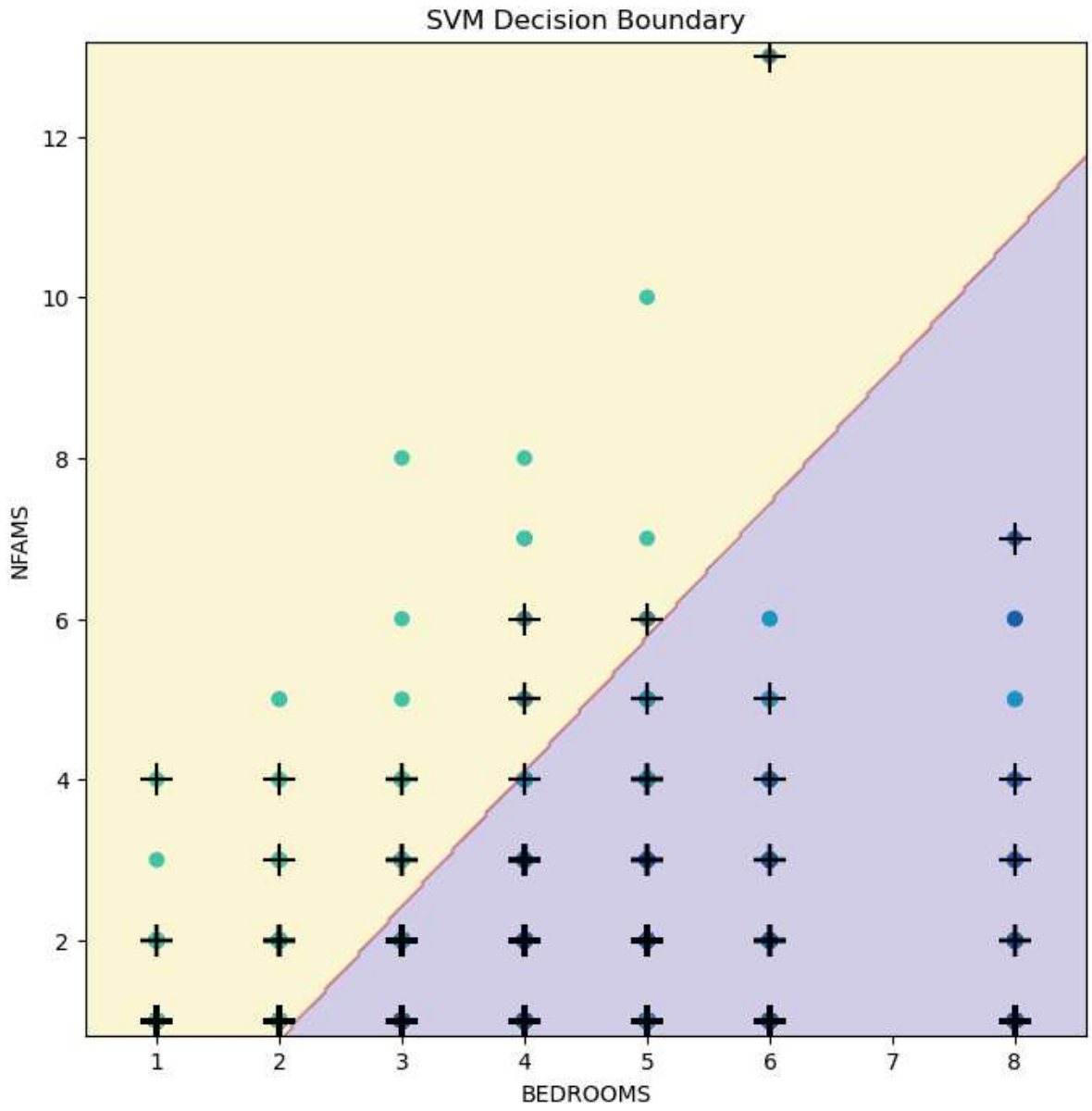
```
                Feature  Coefficient  Absolute_Coefficient
1             BEDROOMS    -0.593302              0.593302
16   Primary_Education     0.362950              0.362950
3                NFAMS     0.354441              0.354441
9         Low_HHINCOME     0.353145              0.353145
18             College    -0.277083              0.277083
12             Married    -0.275316              0.275316
11       High_HHINCOME    -0.272779              0.272779
6          Low_DENSITY    -0.262100              0.262100
8         High_DENSITY     0.187440              0.187440
14              Single     0.153297              0.153297
17  Secondary_Education    -0.144805              0.144805
13           Seperated     0.122019              0.122019
0                ROOMS    -0.112564              0.112564
10     Medium_HHINCOME    -0.080366              0.080366
7       Medium_DENSITY     0.074659              0.074659
2             VEHICLES     0.063603              0.063603
15        No_Education     0.058937              0.058937
5                  AGE    -0.028140              0.028140
4             NCOUPLES    -0.027238              0.027238
```

In [10]:
```
BEDROOMS = X.columns.get_loc('BEDROOMS')
NFAMS = X.columns.get_loc('NFAMS')
fig, ax = subplots(figsize=(8,8))
plot_svm(X,
         y,
         svm_linear,features=(BEDROOMS, NFAMS),
         ax=ax)

plt.xlabel('BEDROOMS')
plt.ylabel('NFAMS')
plt.title('SVM Decision Boundary')

plt.show()
```

```
C:\Users\neela\anaconda3\lib\site-packages\sklearn\base.py:493: UserWarning: X doe
s not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```



Radial

In [26]:
```
svm_rbf = SVC(C=1, kernel="rbf", gamma=1)
svm_rbf.fit(X_train, y_train)
```

Out[26]:
```
SVC(C=1, gamma=1)
```

In [27]:
```
accuracy_rbf_train = svm_rbf.score(X_train, y_train)
accuracy_rbf_train
```

Out[27]:
```
0.9606697277491768
```

In [28]:
```
accuracy_rbf_test = svm_rbf.score(X_test, y_test)
accuracy_rbf_test
```

Out[28]:
```
0.7755654149983768
```

In [29]:
```
param_grid = {'C': [0.01, 0.1, 1, 10],
              'gamma': [0.01, 0.1, 1, 10]}

svm_rbf = SVC(kernel='rbf')
```

```
grid_search = GridSearchCV(estimator=svm_rbf, param_grid=param_grid, cv=5, scoring=
grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
print("Best Parameters:", best_params)

best_score = grid_search.best_score_
print("Best Mean Cross-validated Score:", best_score)
```

```
Best Parameters: {'C': 10, 'gamma': 0.01}
Best Mean Cross-validated Score: 0.8305737131604112
```

In [34]:
```
svm_rbf = SVC(C=10, kernel="rbf", gamma=0.01)
svm_rbf.fit(X_train, y_train)
```

Out[34]:
```
SVC
SVC(C=10, gamma=0.01)
```

In [35]:
```
accuracy_rbf_train = svm_rbf.score(X_train, y_train)
accuracy_rbf_train
```

Out[35]:
```
0.837391586661101
```

In [36]:
```
accuracy_rbf_test = svm_rbf.score(X_test, y_test)
accuracy_rbf_test
```

Out[36]:
```
0.8348663564549291
```

Feature Importance for Radial model

In [6]:
```
svm_radial = SVC(C=10, kernel="rbf", gamma=0.01)
svm_radial.fit(X_train, y_train)

importance_radial = permutation_importance(svm_radial, X_test, y_test, n_jobs=-1)
importance_scores_radial = importance_radial.importances_mean
feature_names_radial = X_train.columns

sorted_indices_radial = importance_scores_radial.argsort()[::-1]
sorted_feature_names_radial = feature_names_radial[sorted_indices_radial]
sorted_importance_scores_radial = importance_scores_radial[sorted_indices_radial]

for feature_name, importance_score in zip(sorted_feature_names_radial, sorted_impor
    print(f"Feature: {feature_name}, Importance Score: {importance_score}")
```

```
Feature: AGE, Importance Score: 0.05495076290444756
Feature: BEDROOMS, Importance Score: 0.05412834108862676
Feature: ROOMS, Importance Score: 0.029542257331457655
Feature: VEHICLES, Importance Score: 0.013786386754680224
Feature: NFAMS, Importance Score: 0.0032896872632832254
Feature: College, Importance Score: 0.0024023374093713025
Feature: Married, Importance Score: 0.0011687046856400762
Feature: Low_HHINCOME, Importance Score: 0.0011470620062763893
Feature: Primary_Education, Importance Score: 0.0004328535872741179
Feature: Medium_HHINCOME, Importance Score: 0.000324640190455594
Feature: NCOUPLES, Importance Score: 0.0002380694730007793
Feature: No_Education, Importance Score: 0.0001514987555459646
Feature: High_HHINCOME, Importance Score: 0.00015149875554592017
Feature: Low_DENSITY, Importance Score: 0.00012985607618223315
Feature: Secondary_Education, Importance Score: 0.0
Feature: Single, Importance Score: -2.1642679363709227e-05
Feature: Seperated, Importance Score: -0.00017314143490962942
Feature: Medium_DENSITY, Importance Score: -0.0002597121523644414
Feature: High_DENSITY, Importance Score: -0.0005410669840926197
```

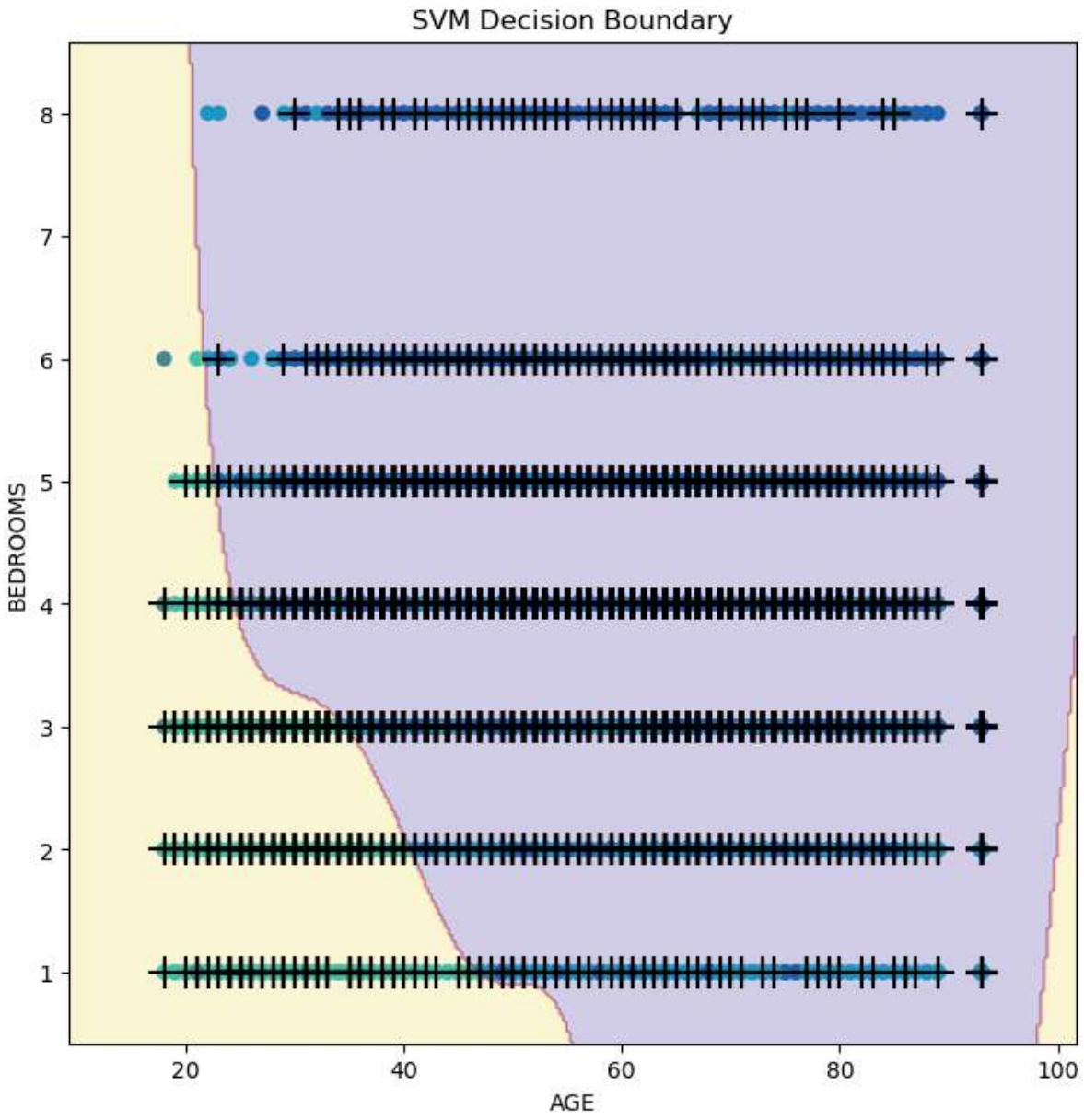
In [11]:

```
AGE = X.columns.get_loc('AGE')
BEDROOMS = X.columns.get_loc('BEDROOMS')
fig, ax = subplots(figsize=(8,8))
plot_svm(X,
         y,
         svm_radial,features=(AGE, BEDROOMS),
         ax=ax)

plt.xlabel('AGE')
plt.ylabel('BEDROOMS')
plt.title('SVM Decision Boundary')

plt.show()
```

```
C:\Users\neela\anaconda3\lib\site-packages\sklearn\base.py:493: UserWarning: X doe
s not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```



Polynomial

In [7]:
```
svm_poly = SVC(C=1, kernel='poly', degree=2)
svm_poly.fit(X_train, y_train)
```

Out[7]:

SVC

SVC(C=1, degree=2, kernel='poly')

In [8]:
```
accuracy_poly_train = svm_poly.score(X_train, y_train)
accuracy_poly_train
```

Out[8]:
```
0.8227354946431056
```

In [9]:
```
accuracy_poly_test = svm_poly.score(X_test, y_test)
accuracy_poly_test
```

Out[9]:
```
0.8297803268044583
```

In [40]:
```
param_grid = {'C': [0.01, 0.1, 1, 10],
              'degree': [2, 3, 4, 5]}

svm_poly = SVC(kernel='poly')
```

```
grid_search = GridSearchCV(estimator=svm_poly, param_grid=param_grid, cv=5, scoring
grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
print("Best Parameters:", best_params)
best_score = grid_search.best_score_
print("Best Mean Cross-validated Score:", best_score)
```

```
Best Parameters: {'C': 10, 'degree': 2}
Best Mean Cross-validated Score: 0.8238021393663804
```

In [12]:
```
svm_poly = SVC(C=10, kernel='poly', degree=2)
svm_poly.fit(X_train, y_train)
```

Out[12]:
```
SVC
SVC(C=10, degree=2, kernel='poly')
```

In [11]:
```
accuracy_poly_train = svm_poly.score(X_train, y_train)
accuracy_poly_train
```

Out[11]:
```
0.8238022355178332
```

In [12]:
```
accuracy_poly_test = svm_poly.score(X_test, y_test)
accuracy_poly_test
```

Out[12]:
```
0.8307542473758252
```

Feature Importance for Polynomial model

In [6]:
```
svm_poly = SVC(C=10, kernel='poly', degree=2)
svm_poly.fit(X_train, y_train)

importance_poly = permutation_importance(svm_poly, X_test, y_test, n_jobs=-1)
importance_scores_poly = importance_poly.importances_mean
feature_names_poly = X_train.columns

sorted_indices_poly = importance_scores_poly.argsort()[::-1]
sorted_feature_names_poly = feature_names_poly[sorted_indices_poly]
sorted_importance_scores_poly = importance_scores_poly[sorted_indices_poly]

for feature_name, importance_score in zip(sorted_feature_names_poly, sorted_importa
    print(f"Feature: {feature_name}, Importance Score: {importance_score}")
```

```
Feature: BEDROOMS, Importance Score: 0.06129206795801321
Feature: AGE, Importance Score: 0.047137755654150015
Feature: ROOMS, Importance Score: 0.014587165891137333
Feature: College, Importance Score: 0.00428525051401365
Feature: VEHICLES, Importance Score: 0.002878476355372839
Feature: Married, Importance Score: 0.0026187642030083723
Feature: Low_HHINCOME, Importance Score: 0.00246726544746243
Feature: High_HHINCOME, Importance Score: 0.0013851314792771685
Feature: NFAMS, Importance Score: 0.0013634887999134592
Feature: Low_DENSITY, Importance Score: 0.0011470620062764114
Feature: Secondary_Education, Importance Score: 0.0008657071745482358
Feature: Medium_HHINCOME, Importance Score: 0.0007791364570934212
Feature: Medium_DENSITY, Importance Score: 0.0005410669840926197
Feature: Seperated, Importance Score: 0.0004761389460015364
Feature: NCOUPLES, Importance Score: 0.00045449626663782714
Feature: Primary_Education, Importance Score: 0.0001514987555459646
Feature: No_Education, Importance Score: -2.1642679363687024e-05
Feature: High_DENSITY, Importance Score: -4.328535872739625e-05
Feature: Single, Importance Score: -0.0009739205713667376
```

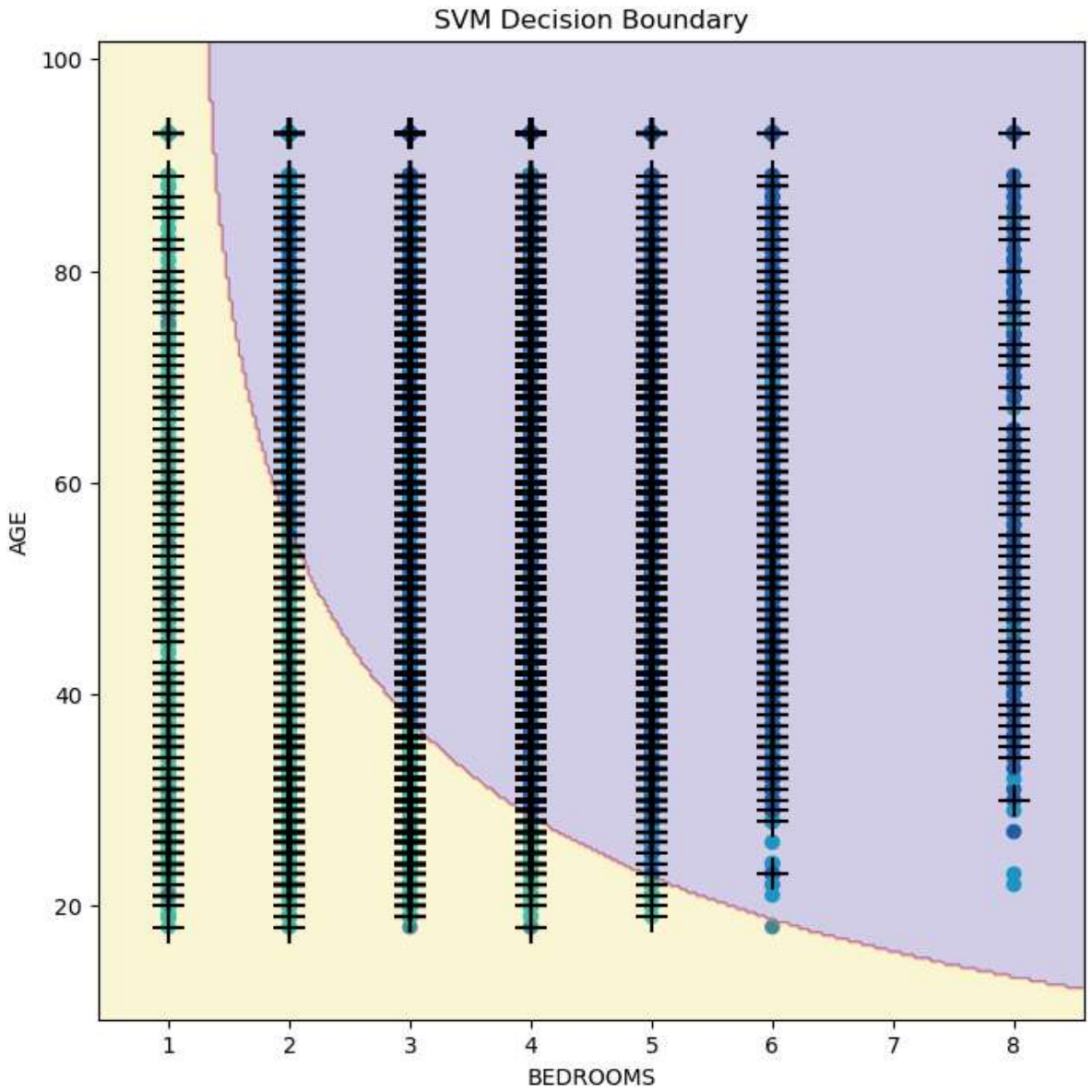
In [13]:

```
BEDROOMS = X.columns.get_loc('BEDROOMS')
AGE = X.columns.get_loc('AGE')
fig, ax = subplots(figsize=(8,8))
plot_svm(X,
         y,
         svm_poly,features=(BEDROOMS, AGE),
         ax=ax)

plt.xlabel('BEDROOMS')
plt.ylabel('AGE')
plt.title('SVM Decision Boundary')

plt.show()
```

```
C:\Users\neela\anaconda3\lib\site-packages\sklearn\base.py:493: UserWarning: X doe
s not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```



In [ ]: