Analysis On Dwelling Occupied By Owners Or Renters

Name: Lakshit Gupta
Course: DATA 5322-01

Abstract

This report presents an in-depth analysis of Support Vector Machines (SVMs) for classification tasks applied to a dataset concerning ownership status (OWNERSHP) prediction. The study uses various SVM configurations with different kernel functions, including linear, radial and polynomial kernels, to detect the impact of different parameters on model performance and feature importance. This study also takes into account the socioeconomic factors as well as the demographic factors, and highlights the main predictors that can influence the ownership of the house like number of rooms in the house, age of the person living in the house, marital status of the person living in the house and many more. Decision boundaries were also visualized for a better understanding of the model's classification patterns. For the Linear kernel SVM we achieved an accuracy of 84.11% with our top 10 important predictors, the radial SVM got the accuracy of 84.90% while polynomial SVM was the 2nd highest with an accuracy of 84.23%. The results also help to get a broader perspective of SVM's use in real world classification tasks and provide important insights for practical implementations.

Introduction

The aim of this project is to use Support Vector Machine (SVM) classification models to analyze a large dataset obtained from Washington State. The data comes from the US Census via IPUMS USA [1] and has 75,388 observations with 24 variables, including demographics, housing details, and socioeconomic indicators. The goal is to predict whether a dwelling is occupied by owners or renters by taking OWNERSHP as our target variable where 1 means the dwelling is occupied by the home owners whereas 2 means the dwelling is occupied by the renters. By exploring the relationships between factors like age, income, education, housing costs, and population density, we hope to gain insights into the factors influencing homeownership.

We're using different types of SVM models like linear, radial, and polynomial kernels, to see which one better predicts the homeownership in Washington State. These models help us understand how different factors relate to whether someone owns or rents their home.

Theoretical Background

- **Support Vector Machine:**

  Support vector machine (SVM) is a supervised learning algorithm which is used for classification and regression problems. It is an effective classifier that can be used to solve linear problems. SVM also supports kernel methods to handle nonlinearity. Given training data, the idea of SVM is that the algorithm creates a line or a hyperplane which separates the data points into classes. Thus SVM creates a decision boundary which ensures that the separation between the classes is maximum. It tries to maximize the separation of the data set and the decision surface.

  For the kernel function in support vector machines, it can be divided into 3 different categories for this report. Different classes are separated by different hyperplanes, and to find the best hyperplane for the data, kernel functions are used.

- **Linear Kernel:**

  The Linear kernel processes the dot product of input vectors to identify a straight-line decision boundary. It performs better with data sets where categories are divided with clear space in between. The Linear kernel operates smoothly and manages an abundance of characteristics as well. But, if the data doesn't follow a straight line, they can face difficulties. This kernel is the most suitable for data that can be clearly divided into direct linear groups. It performs quickly, making it great for processing large sets of multi-dimensional data. However, for data that doesn't follow a straight path, other kernels may perform better.

- **Radial Kernel:**

  The Radial kernel is another name for the rbf kernel. It uses a gamma function to map data into a higher space with more dimensions. This allows it to work well with data where features have complex, non-linear relationships. RBF kernels are flexible and can capture intricate decision boundaries. But, they need careful tuning of settings like gamma parameter and cost function parameter (C). They can also be computationally costly, especially for large datasets with many data points.

- **Polynomial Kernel:**

  The Polynomial kernel changes data to a space with higher dimensions. It uses polynomial functions for this. The kernel works well when the data has non-linear relationships between features. However, it's not as flexible as the RBF kernel.

Polynomial kernels can understand complex decision boundaries. But if the degree of the polynomial is too high, overfitting may occur. Need to tune hyperparameters like the polynomial's degree and the cost function (C). Overall, polynomial kernels balance computational efficiency and flexibility in capturing non-linear relationships within the data.

- **Regularization Parameter (C) :**

  It controls two things: training error and testing error. A higher C value is great at correct classification during training, but it might overfit. A lower C value makes a wider margin, so it may increase how well it works in new cases. The middle ground between fitting training data perfectly and generalizing well is key.

- **Gamma parameter for RBF kernel :**

  The parameter γ influences the width of the Gaussian function in RBF kernels. A small γ value widens the kernel, leading to smoother decision boundaries. But, a large γ value creates narrow kernels with complex boundaries. Finding the correct balance avoids overfitting or underfitting issues.

- **Degree parameter for polynomial kernel :**

  Polynomial kernels have a degree parameter. A high degree leads to detailed decision boundaries. This lets the model learn complex patterns well, but can result in overfitting if the degree is too high. Lower degrees make simple boundaries that may reduce overfitting risk. But if the degree is extremely low, the model might underfit instead.

Methodology

After looking into the dataset, several steps were taken to get the data ready for the analysis. The dataset was quite large, containing 75,388 entries with 24 columns. However, to ensure the accuracy of the analysis, certain columns that weren't important for the analysis were removed, such as 'OWNERSHPD' which was the detailed version of ownership of the dwelling, 'NCOUPLES' which was the number of couples in the household, 'NFAMS' which was the number of families in the household, ‘PERNUM’ which was the number of people in the household, ‘BIRTHYR’ which was the birth year of the person living in the household, ‘EDUCD’ which was the detailed version of the educational attainment of the person and similar others.

Another variable that was dropped was ‘ValueH’ which was an important variable but needed to be dropped for further analysis of the data. After dropping these columns, the dataset was left with 14 columns and 72,631 entries.

To further correct the data, some rows were filtered with specific values in certain columns such as which was filled with NA and Missing. For example, rows where 'OWNERSHP' was 0, indicating unknown ownership status was removed, and where 'BEDROOMS', 'ROOMS', 'BUILTYR2', 'MARST', and 'VEHICLES' had values of 0 or 9, which were NA or missing data. Additionally, numerical values in the 'MARST' column were replaced with more descriptive labels and divided them into 3 categories, such as 'Married', 'Divorced', and 'Single', to make the data more readable.

Next, the data was sorted by 'SERIAL' which was the unique number for every dwelling and 'AGE' which was the age of the person, then grouped it by 'SERIAL' to keep only the first entry for each 'SERIAL' value as to assign each dwelling to the oldest person living in the house, effectively reducing the dataset to 29,049 entries. Finally, the 'SERIAL' column was removed as it was no longer needed for the analysis.

The resulting dataset consisted of 29,049 entries and 15 columns, including features like 'DENSITY' which was population of the area, 'COSTELEC' which was the cost of electricity, 'COSTGAS' which was the cost of gas, and similar others, which were used to predict homeownership status ('OWNERSHP'). Then the data was split into training and testing sets, applied standard scaling to ensure uniformity within the features, and proceeded with our analysis.

- **Linear Kernel:**

  Exploration was done, how different cost values affect the accuracy of a linear Support Vector Machine (SVM) model. Then used various cost values, ranging from 0.01 to 500 with various intervals, to train SVM models and checked their accuracy on the testing data. By plotting the accuracies against the cost values, aim was to identify the optimal cost value that maximizes the model's performance.
  Next, GridSearchCV was performed to provide me the optimal cost value and also the accuracy of that best model. After that feature selection using SelectKBest[2] was performed with permutation information classification[3] to get the top 10 features from my model. Then selection of the top 2 features was done and trained another SVM model using only these features. After training an SVM model with the selected features, evaluation of its accuracy was done on the testing data. This process helped to identify whether these two features will increase the accuracy of the model and how strong an effect it has only on the target variable.
  Finally, visualization of the decision boundary of the SVM model using the top two predictor variables. This decision boundary helps to understand how the model distinguishes between owned and rented homes based on these two top features.

- **Radial Kernel:**

  Proceeding with the SVM analysis with a radial kernel by performing GridSearchCV to explore how different cost and gamma values affect the accuracy of a radial Support Vector Machine (SVM) model. Various cost values were used, ranging from 0.01 to 1000 with various intervals while gamma values ranging from 0.5 to 4 with various intervals to train SVM models and check their accuracy on the testing data. By plotting the accuracies against the different cost and gamma values, aimed to identify the optimal parameter values that maximizes the model's performance.
  After that feature selection using SelectKBest[2] was performed with permutation information classification[3] to get the top 10 features from the model. Then selection of the top 2 features and training another SVM model using only these features. After training an SVM model with the selected features, evaluation of its accuracy on the testing data was done. This process helped to identify whether these two features will increase the accuracy of the model and how strong an effect it has only on the target variable.
  Finally, visualization of the decision boundary of the SVM model using the top two predictor variables. This decision boundary helps to understand how the model distinguishes between owned and rented homes based on these two top features.

- **Polynomial Kernel:**

  Ending the SVM analysis with polynomial kernel by performing GridSearchCV to explore how different cost and degree values affect the accuracy of a polynomial Support Vector Machine (SVM) model. Various cost values were used, ranging from 0.01 to 1000 with various intervals while degree values ranging from 2 to 4 to train SVM models and check their accuracy on the testing data. By plotting the accuracies against the different cost and degree values, aim to identify the optimal parameter values that maximizes the model's performance.
  After that feature selection using SelectKBest[2] was performed with permutation information classification[3] to get the top 10 features from the model. Then selection of the top 2 features and training another SVM model using only these features. After training an SVM model with the selected features, evaluation of its accuracy on the testing data. This process helps to identify whether these two features will increase the accuracy of the model and how strong an effect it has only on the target variable.
  Finally, visualization of the decision boundary of the SVM model using the top two predictor variables. This decision boundary helps us understand how the model distinguishes between owned and rented homes based on these two top features.

Computational Results

- **Linear Kernel**

  For performing the SVM with a linear kernel, implementation of the model with different cost values was done to check how it affects the accuracy of our SVM model. Then validated that by performing GridSearchCV that also gave almost the same results.

Cost	Accuracy
0.01	**84.28%**
0.1	**84.31%**
1	**84.30%**
10	**84.30%**
100	**84.29%**
250	**84.31%**
500	**84.54%**

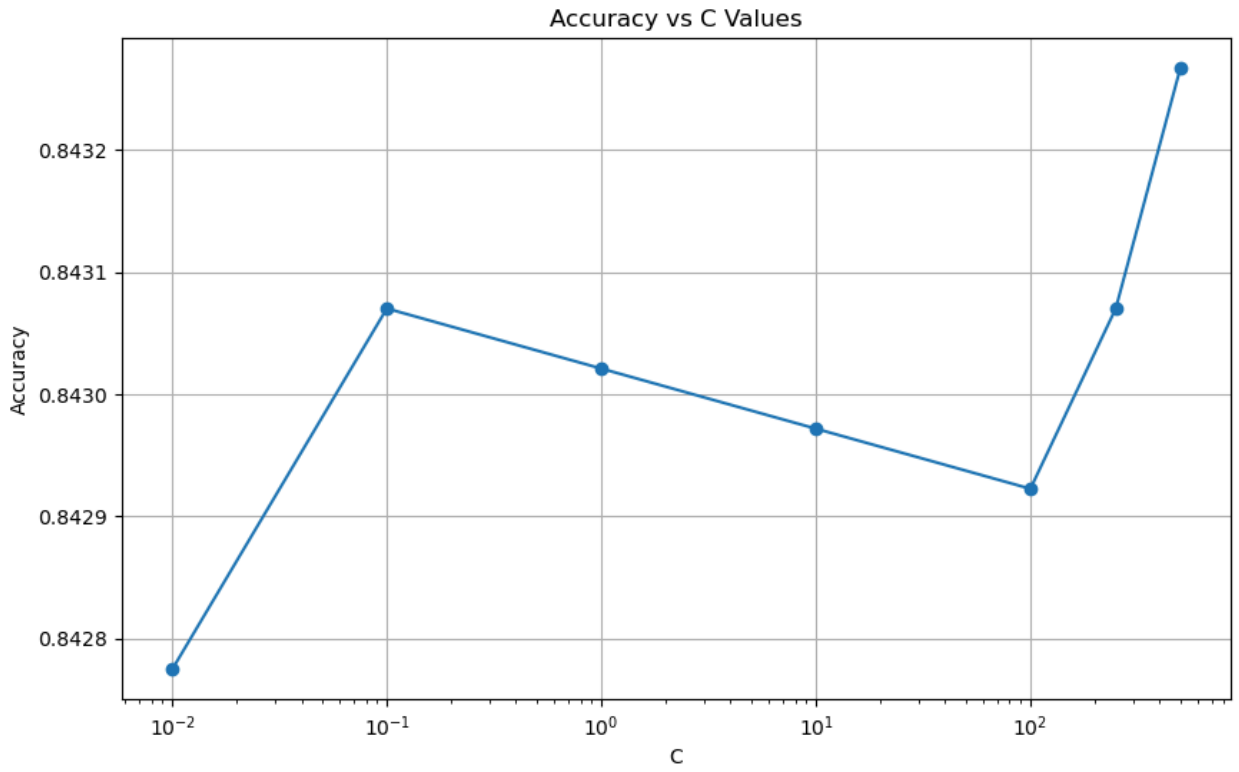

Figure 1. Plot of Accuracy vs Cost Values

From the above plot figure 1. and the table, it can be observed that the model performed the best at an accuracy of 84.54% with a cost of 500. And it is not much of a difference in the accuracy with the other costs as well. For the cost values at 0.1,1,10,100 and 250 it is giving almost the same accuracy of 84.30%

After performing the feature selection and with those top 10 features that were selected, the model achieved an accuracy of 84.11%, which is less than the model without feature selection.

- **Radial Kernel**

Testing of SVM with a radial kernel was done using different cost and gamma values to see how it affects the model's accuracy. GridSearchCV was used to double-check and to find out the best parameters values for the model.

Used various cost values, ranging from 0.01 to 1000 with various intervals while gamma values ranging from 0.5 to 4. The optimal value cost was 1 and the optimal gamma value was 0.5. With these parameter values an accuracy of 84.69% was achieved.

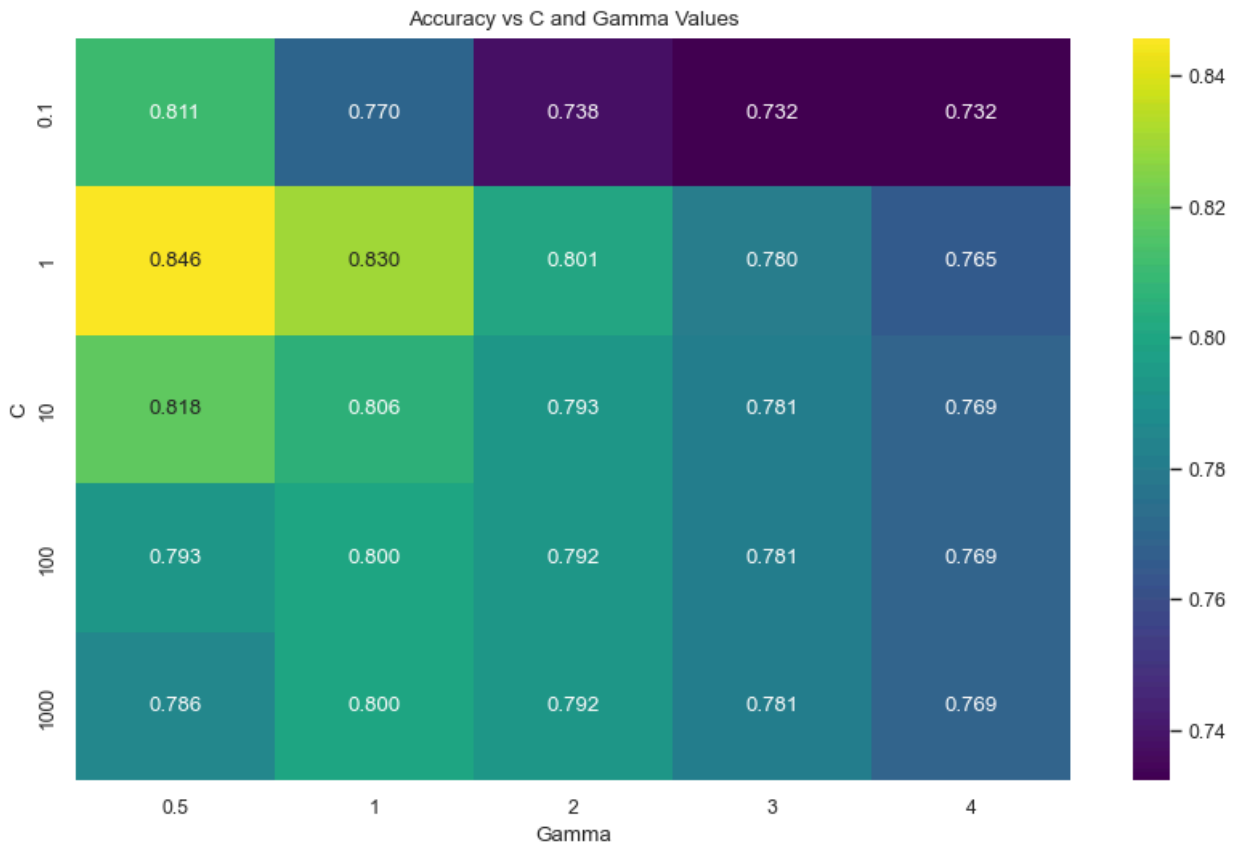

Figure 2. Heatmap for Accuracy vs C and Gamma Values before feature selection

The above plot shows how the model performed for various cost and gamma values and what accuracy the model achieved with those specific values.

After the feature selection with the top 10 variables an increased accuracy of 84.90% was achieved.

- **Polynomial Kernel**

For the SVM with polynomial kernel, processing with multiple values of cost as well as multiple values of degree was done. With that an accuracy of around 85.19% was achieved.

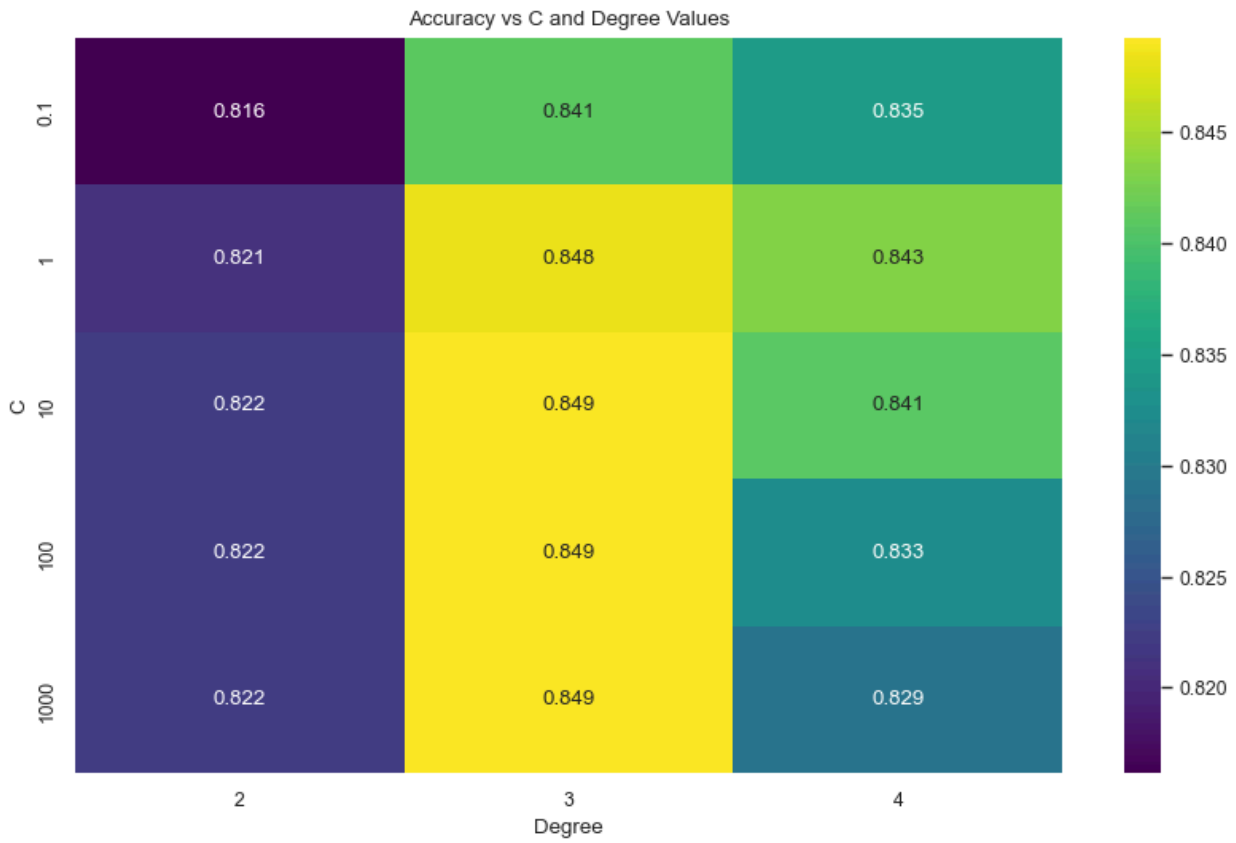

Figure 3. Heatmap for Accuracy vs C and Degree Values before feature selection

After taking the most important variables into account through feature selection and training the model on those top 10 variables, an accuracy of 84.23% was achieved which is less than the model on which feature selection was not performed.

Discussion

- **Preprocessing**

  Deciding not to use the "ValueH" variable which was the value of the house in our analysis was important, even though it seemed like it could really help predict if someone owns or rents their home. Using it might have made our model seem really good as it was giving us an accuracy of 100%, but it could also have caused some problems. It might have made the model focus too much on just one thing and not consider other important variables.

- **Linear kernel**

The linear kernel showed constant performance across different cost values, with accuracy ranging from 84.28% to 84.54%. GridSearchCV confirmed that the best cost value was 500, resulting in an accuracy of 84.54%.

Performing feature selection with the best cost value was necessary to get the top 10 features so as to check what all variables are the most influential on detection whether the owner is residing in the dwelling or it has been rented.

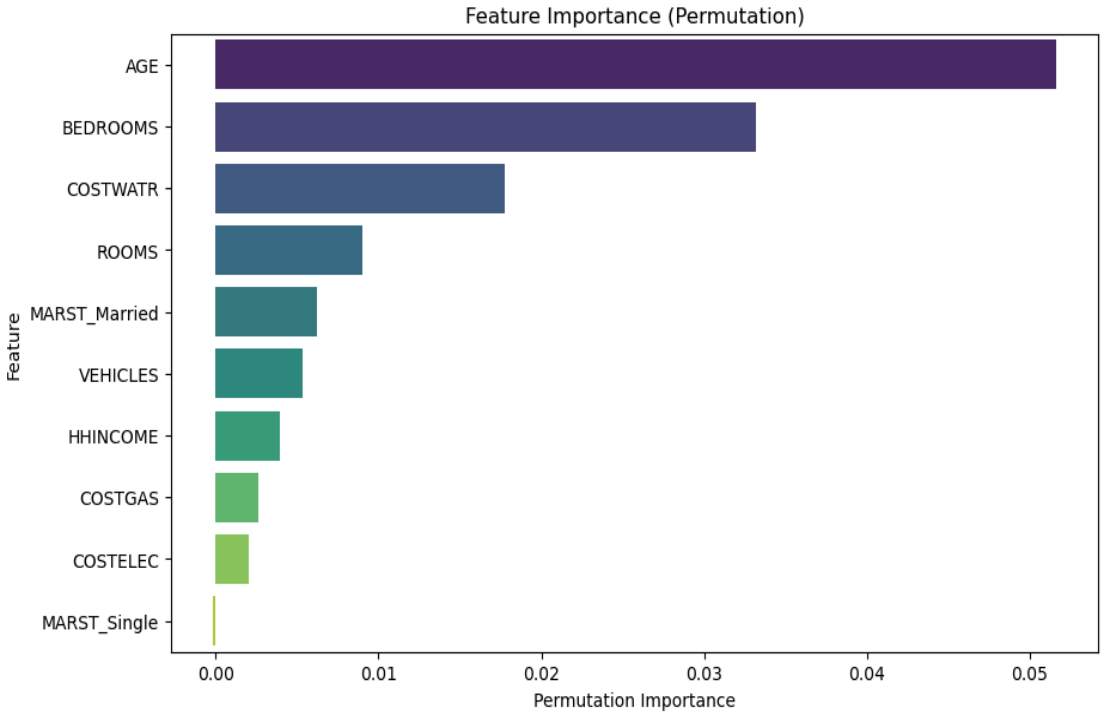

Figure 4. Feature importance of top 10 variables

From the above plot it can be seen that the age, number of bedrooms, cost of water , number of rooms and marital status of the person specifically being married can be influential in predicting that the house is owned or rented.

Age can be an influential variable as younger individuals tend to rent out more than buying the house. Also the number of bedrooms can be one of the important factors as it tells about the functionality of the house and most families buy houses based on this factor. After performing feature selection and then only taking the top 10 variables decreased our accuracy of the model to 84.11%, which suggests that not only these variables are important. The other variables are also somewhat influential in predicting homeownership.

Variables like density of the place where the house is can be important as people usually buy homes which are nearer to the city areas for easier commute.

Older people with bigger homes including more bedrooms and rooms, lower water costs, and being married makes it likely to own their homes. This shows stability, finance, and family status matters greatly for the ownership of the house. The results also tell us about the obstacles numerous people face affording a home. To aid more people in becoming homeowners, recommendations could get forwarded towards decision makers in housing organizations.

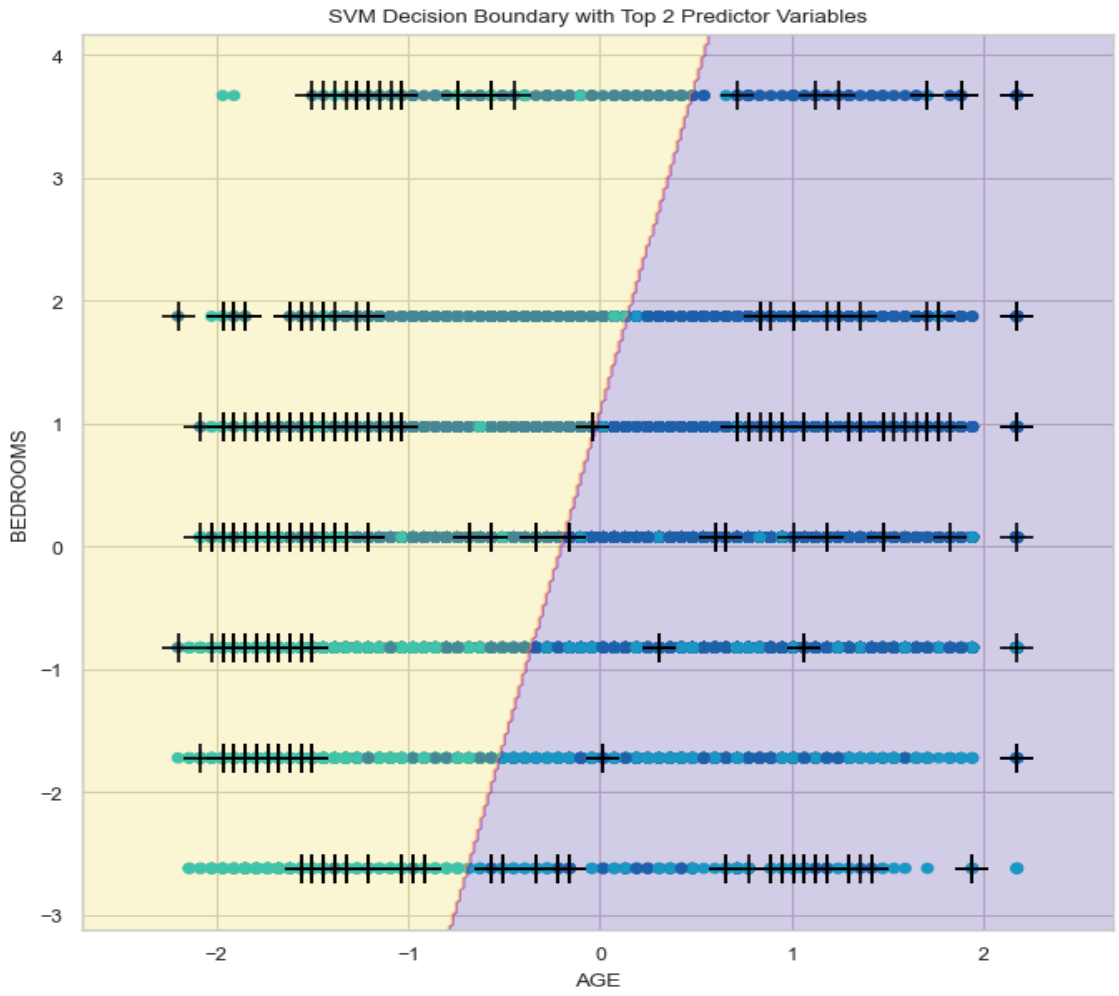

Figure 5. Decision boundary plot for Age VS Bedrooms

The purple area toward the lower left corner of the decision curve contains a dense concentration of '+' symbols, indicating that younger people with fewer bedrooms are more likely to be renters. The dense groups of '+' symbols in the yellow region towards the top right corner of the decision curve indicates that older people who have more bedrooms are more likely to be owners. There are particular points of data that seem to be incorrectly classified or to be near the boundary of the decision, showing possible errors or overlaps in the data.

Age and bedroom were our top 2 variables and performing the model on it reduced our models accuracy by 30% which is a lot, suggesting that apart from these variables, the model requires additional features for maintaining optimal accuracy.

- **Radial kernel**

With various values of cost and gamma, while performing GridSearchCV at cross validation value of 5, the accuracy received by the SVM with Radial kernel was 84.69%. The optimal value of cost was 1 and for gamma it was 0.5.

Next, performing feature selection with the optimal parameters and getting the top 10 variables. After this trained a SVM model with a radial kernel but now only with those top variables, the model achieved an accuracy of 84.90% which is more than what it was achieved for the model without feature selection.

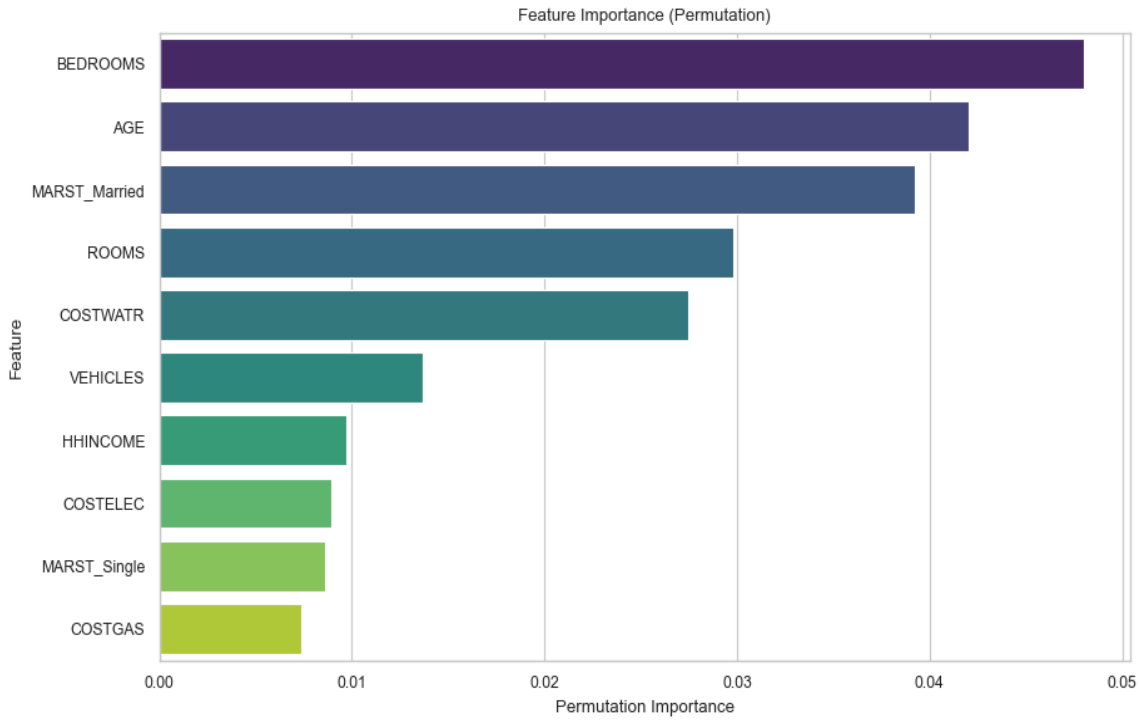

Figure 6. Feature importance of top 10 variables

Number of bedrooms, age, marital status as married , number of rooms and cost of water comes out to be the important variables. The variables are the same as what it was there for the linear kernel but the importance level of these are different than earlier.

Over time, people's choices for purchasing homes changes. Age used to be the main determinant. Our radial SVM model suggests that the quantity of bedrooms is more important. This change implies that logical factors, like the size of a home, should come first. Decisions are also influenced by factors like water expenses and marital status.

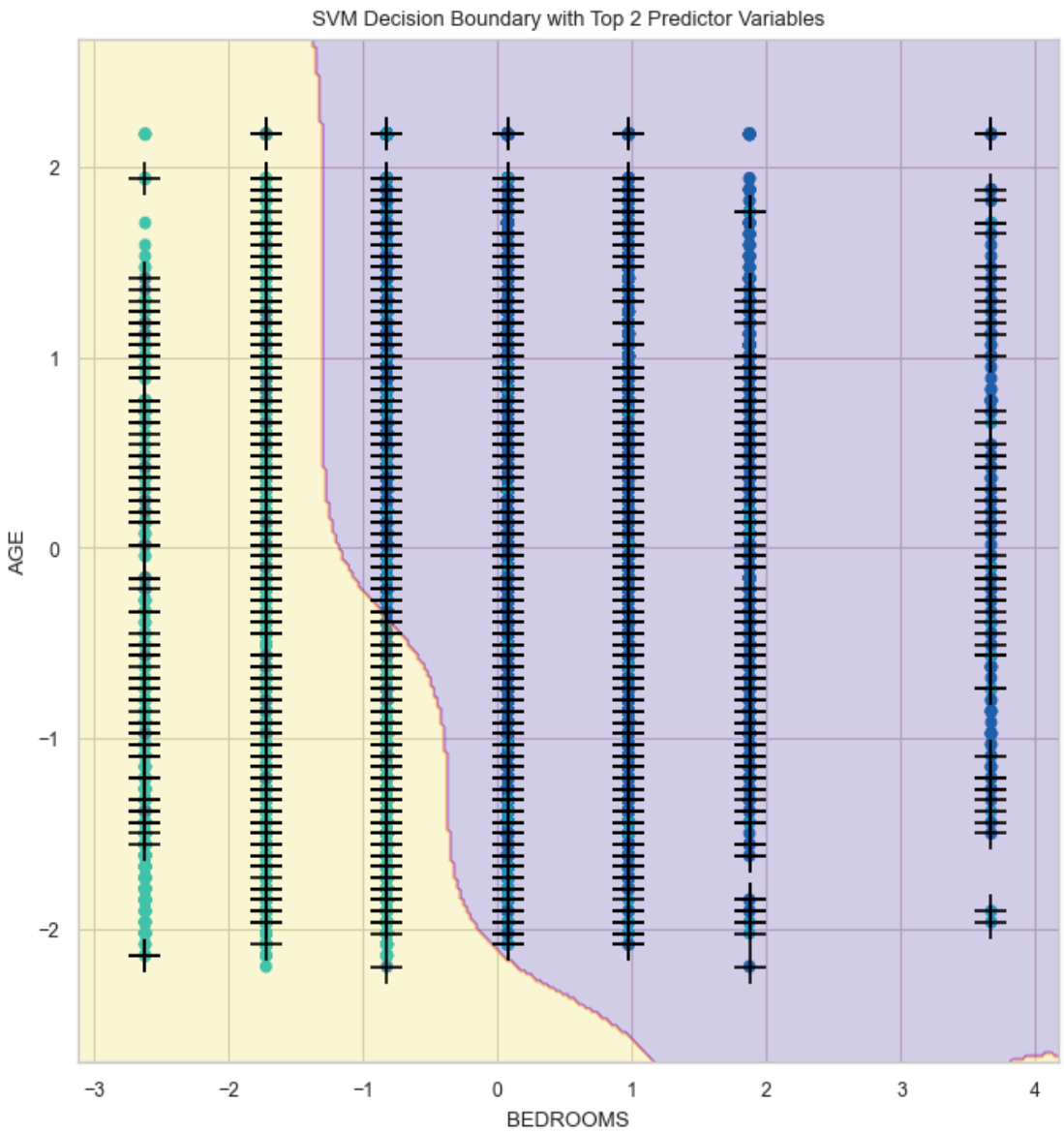

Figure 7. Decision boundary plot for Bedrooms VS Age

It is clear that the boundary line that divides owners from renters is curved rather than straight. This is as a result of the usage of radial kernel SVM. The x-axis represents the number of bedrooms, while the y-axis represents the age variable. The plot is divided into two regions: the yellow region on the left, and the purple region on the right, separated by the curved decision boundary line. Data points (represented by '+') that fall in the yellow region are predicted by the model to be occupied by owners, while data points in the purple region are predicted to be occupied by renters. Renters typically live in homes with fewer bedrooms and younger occupants; this is shown by the dense concentration of '+' symbols in the purple area toward the lower left corner of the decision boundary.

Owner occupancy is more common in homes with more bedrooms and older residents, as shown by the dense '+' symbol clustering in the yellow region towards the upper right corner of the decision boundary.
The non-linear relationship between the number of bedrooms and age is reflected in the non-linear, curved decision boundary line that divides the two classes.

- **Polynomial kernel**

GridSearchCV showed that a polynomial degree of 3, and a 'C' value of 100 were the optimal parameters for the SVM model with a polynomial kernel. The test data showed that this particular combination produced an accuracy of 85.19%, which shows how well the model worked to predict whether people owned or rented their homes. However, the accuracy dropped to 84.23% when only a few features were taken into account.

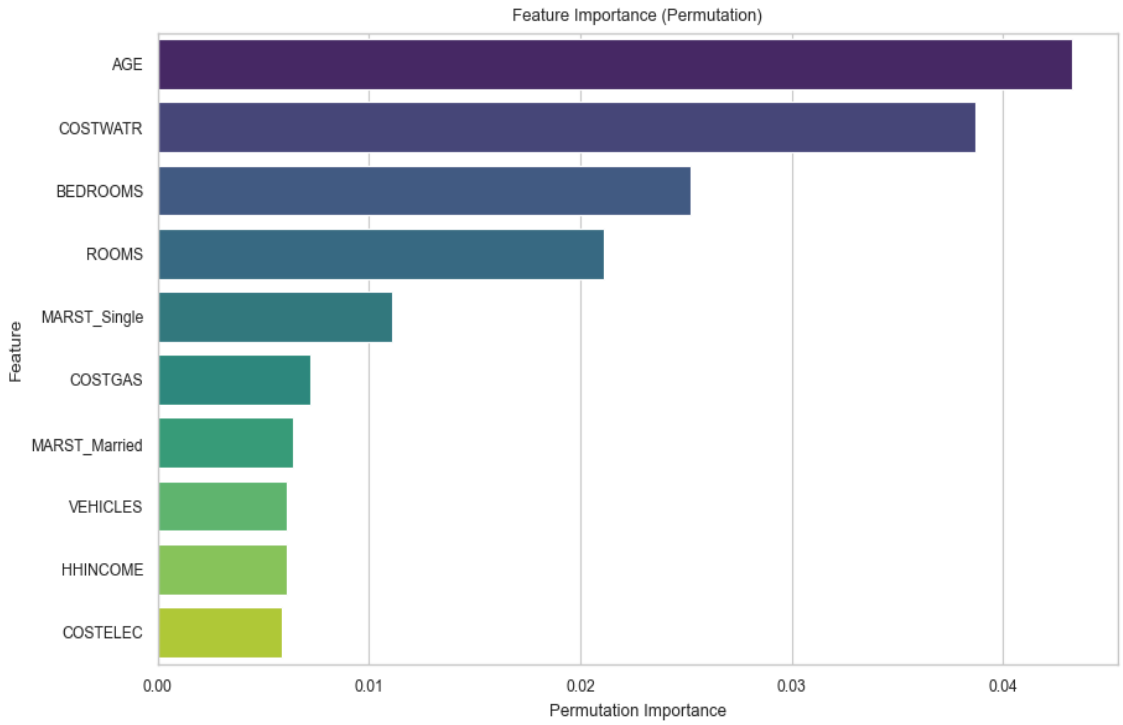

Figure 8. Feature importance of top 10 variables

Some more results were obtained by analyzing the importance of features in predicting homeownership. The two important variables were found to be age and the cost of water, showing that these factors have an important effect on the choices of individuals to buy or rent a home. The number of bedrooms and rooms also had a significant impact, showing that a home's size and useful characteristics have an impact on those who choose to become homeowners.

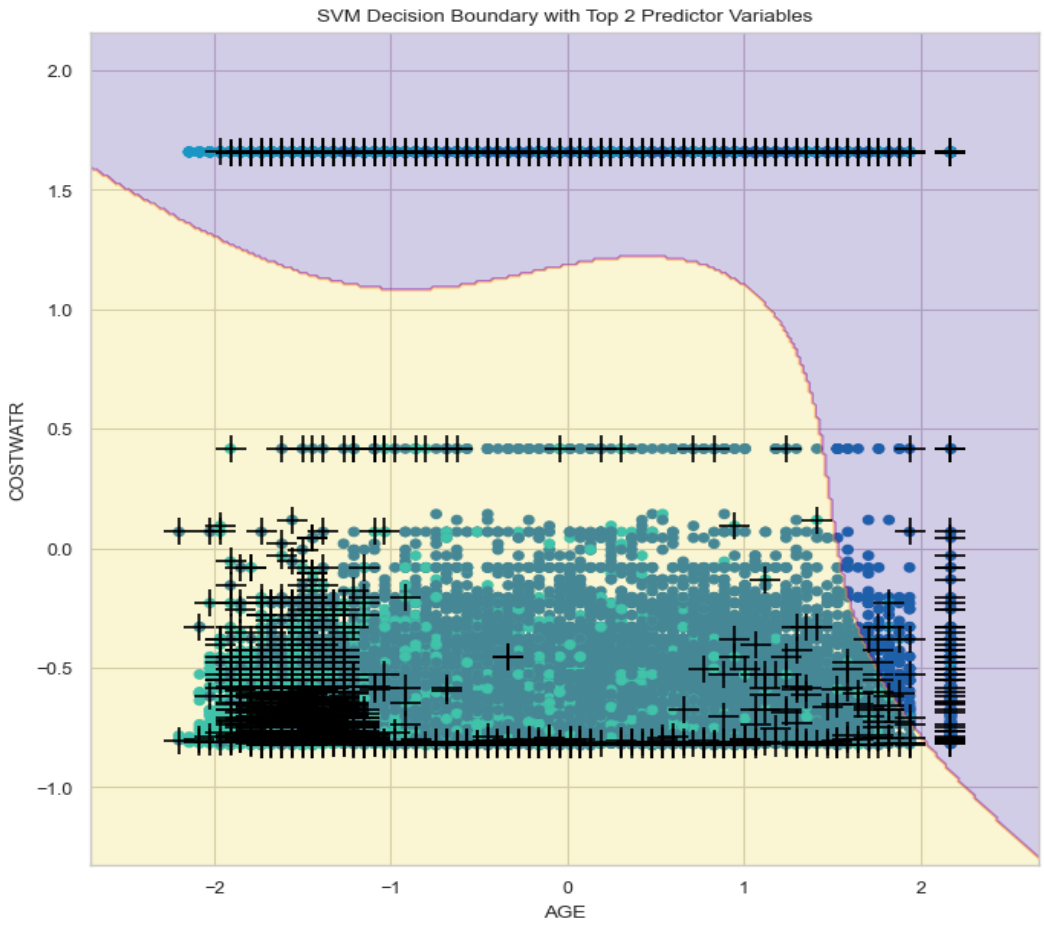

Figure 9. Decision boundary plot for Age VS COSTWATR

From the above plot we can see that the x-axis represents the age variable, while the y-axis represents the COSTWATR variable. The plot is divided into two regions: the yellow region and the purple region, separated by a complex, curved decision boundary line.

The data points that are predicted to be renters seem to be concentrated at lower COSTWATR values and younger ages, indicating that renters are typically younger people with lower utility costs.
The dense clustering of '+' symbols in the yellow region indicates that the chance of the dwelling being occupied by owners increases with age and COSTWATR values.
The decision boundary line, which reflects the complex, polynomial relationship between COSTWATR and age. It divides the two classes in a highly non-linear and curved manner.
There are multiple data points that seem to be incorrectly classified or to be near the boundary of the decision, suggesting misclassification and overlaps in the data.

Conclusion

In conclusion, our study used Support Vector Machines (SVMs) with various kernel functions to determine the variables and predictors influencing the prediction of homeownership. We looked at a number of socioeconomic and demographic variables, including age, number of bedrooms, cost of water, number of rooms, and marital status. With various SVM kernels, including linear, radial, and polynomial kernels, some of the predictors were consistently present as influential factors in distinguishing between homeowners and renters.

Age of the person living in the household came out to be the one of the most influential predictor variables. It could be correct as older people usually tend to own their own homes, while most of the young generation usually rents the place.

Number of bedrooms also took the spot in the top two predictor variables, as families usually buy those houses which have more bedrooms in them.

Apart from these two other predictors are also important in determining if the house is occupied by owners or renters as when we fit the model only using these two top predictors, the accuracy of the model dropped for all the kernel SVM models. So it suggests to us that other variables are also required to give more information to the model to process on.

Our SVM model with Radial kernel out performed both the model with linear kernel and the model with polynomial kernel as it distinguishes better between the owners and the renters, with an accuracy of 84.69% with all the variables and also the accuracy increased when we only took the top 10 variables to 84.90%. These results tell us about the effectiveness of the Radial kernel in accurately predicting homeownership status, showing its ability to capture the complexities of the dataset with a non linear boundary and identify the most relevant predictors.

References

- https://www.ipums.org/projects/ipums-usa/d010.V13.0
- https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#
- https://scikit-learn.org/stable/modules/permutation_importance.html

Appendix

Preprocessing

```
# Import libraries
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib.pyplot import subplots, cm
from ISLP.svm import plot as plot_svm
from sklearn.svm import SVC
from mlxtend.plotting import plot_decision_regions
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.inspection import permutation_importance
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.feature_selection import SelectKBest,mutual_info_classif
from sklearn.pipeline import Pipeline
from sklearn.metrics import confusion_matrix
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")


data=pd.read_csv('/Users/lakshitgupta/Library/CloudStorage/OneDrive-SeattleUniversity/Quater3/Machine Learning-2/Written Homeworks/Housing.csv')


print(data.head())


data.shape


missing_values = data.isna().sum().sum()
print(f"Number of missing values: {missing_values}")


data = data.drop(['OWNERSHPD','NCOUPLES','NFAMS','PERNUM', 'PERWT', 'BIRTHYR','EDUC','EDUCD','INCTOT','VALUEH'], axis=1)
```

```
data = data[~data['OWNERSHP'].isin([0])]
data = data[~data['BEDROOMS'].isin([00])]
data = data[~data['ROOMS'].isin([00])]
data = data[~data['BUILTYR2'].isin([00])]
data = data[~data['MARST'].isin([9])]
data = data[~data['VEHICLES'].isin([0,9])]
data = data[~data['AGE'].isin([999])]

print(data.shape)


data['MARST'] = data['MARST'].replace({1: 'Married', 2: 'Married', 3: 'Divorced', 4: 'Divorced', 5:
'Single', 6: 'Single'})
data = pd.get_dummies(data, columns=['MARST'])
print(data.columns)


print(data.head())


data_sorted = data.sort_values(['SERIAL', 'AGE'], ascending=[True, False])
data_sorted = data_sorted.groupby('SERIAL').first().reset_index()
data = data_sorted.drop(columns=['SERIAL'])
data.shape


data.columns


X = data[['DENSITY', 'COSTELEC', 'COSTGAS', 'COSTWATR', 'COSTFUEL',
        'HHINCOME', 'ROOMS', 'BUILTYR2', 'BEDROOMS',
        'VEHICLES', 'AGE', 'MARST_Divorced', 'MARST_Married', 'MARST_Single']]
y = data['OWNERSHP']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=123)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Linear model

```
cost_values = [0.01, 0.1, 1, 10, 100, 250, 500]

costs = []
accuracies = []
```

```
for c in cost_values:
    # Initialize the SVM classifier with the specified cost value
    svm_classifier = SVC(kernel='linear', C=c)

    try:

        svm_classifier.fit(X_train_scaled, y_train)
        y_pred = svm_classifier.predict(X_test_scaled)
        accuracy = accuracy_score(y_test, y_pred)
        costs.append(c)
        accuracies.append(accuracy)

        print(f"Cost: {c}, Accuracy: {accuracy*100:.2f}%")
    except Exception as e:
        print(f"Error occurred for cost {c}: {e}")

plt.figure(figsize=(10, 6))
plt.plot(costs, accuracies, marker='o')
plt.title('Accuracy vs Cost Value')
plt.xlabel('Cost Value')
plt.ylabel('Accuracy')
plt.xscale('log')
plt.grid(True)
plt.show()



param_grid = {'C': [0.01, 0.1, 1, 10, 100, 250, 500]}
costs = param_grid['C']
accuracies = []

grid_search = GridSearchCV(SVC(kernel='linear'), param_grid, cv=5, scoring='accuracy',
n_jobs=-1)
grid_search.fit(X_train_scaled, y_train)

print("Cost values and accuracies for each cost:")
for cost, accuracy in zip(costs, grid_search.cv_results_['mean_test_score']):
    print(f"Cost: {cost}, Accuracy: {accuracy*100:.2f}%")

print("\nBest parameters found by GridSearchCV:")
print(grid_search.best_params_)

# Get the best model from the grid search
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test_scaled)
```

```
accuracy = accuracy_score(y_test, y_pred)

print(f"\nAccuracy of the best model: {accuracy*100:.2f}%")


mean_test_scores = np.array(grid_search.cv_results_['mean_test_score'])

# Plot accuracy against C values
plt.figure(figsize=(10, 6))
plt.plot(param_grid['C'], mean_test_scores, marker='o')
plt.title('Accuracy vs C Values')
plt.xlabel('C')
plt.ylabel('Accuracy')
plt.xscale('log')
plt.grid(True)
plt.show()


selector = SelectKBest(score_func=mutual_info_classif, k=10)
selector.fit(X_train_scaled, y_train)
selected_features = selector.get_support(indices=True)
selected_feature_names = X.columns[selected_features]

svm_selected = SVC(kernel='linear', C=500, random_state=123)
svm_selected.fit(X_train_scaled[:, selected_features], y_train)

accuracy = svm_selected.score(X_test_scaled[:, selected_features], y_test)
print("Accuracy with selected features: {:.2f}%".format(accuracy * 100))

# Permutation importance
perm_importance = permutation_importance(svm_selected, X_test_scaled[:,
selected_features], y_test, n_repeats=30, random_state=123)

# Feature importances
importance_df = pd.DataFrame({'Feature': selected_feature_names, 'Importance':
perm_importance['importances_mean']})
importance_df.sort_values(by='Importance', ascending=False, inplace=True)
print(importance_df)

plt.figure(figsize=(10, 6))
sns.barplot(data=importance_df, x='Importance', y='Feature', palette='viridis')
plt.xlabel('Permutation Importance')
plt.ylabel('Feature')
plt.title('Feature Importance (Permutation)')
plt.show()


X_top = data[['AGE', 'BEDROOMS']]
```

```
y = data['OWNERSHP']
X_trainTop, X_testTop, y_trainTop, y_testTop =  train_test_split(X_top, y, test_size = 0.3,random_state = 123)
svm_classifier = SVC(kernel='linear', C=500, cache_size=1000, verbose=True, max_iter=10000, random_state=123)
svm_classifier.fit(X_trainTop, y_trainTop)
y_pred = svm_classifier.predict(X_testTop)
accuracy = accuracy_score(y_testTop, y_pred)
print(f"Accuracy with top 2 predictors: {accuracy*100:.2f}%")


X_top = data[['AGE', 'BEDROOMS']]

y = data['OWNERSHP']
X_trainTop, X_testTop, y_trainTop, y_testTop =  train_test_split(X_top, y, test_size = 0.3
                                    ,random_state = 123)

scaler = StandardScaler().fit(X_trainTop)
X_trainScaledTop = scaler.transform(X_trainTop)
X_testNewScaledTop = scaler.transform(X_testTop)
linearsvc_top = SVC(kernel='linear', C=500, cache_size=1000, verbose=True, max_iter=10000, random_state=1)
linearsvc_top.fit(X_trainScaledTop, y_trainTop)

fig, ax = subplots(figsize=(8,8))
plot_svm(X_trainScaledTop,
        y_trainTop,
        linearsvc_top,
        ax=ax)
ax.set_xlabel('AGE')
ax.set_ylabel('BEDROOMS')
plt.title('SVM Decision Boundary with Top 2 Predictor Variables')
plt.show()
```

Radial model

```
param_grid = {'C': [0.1, 1, 10, 100, 1000], 'gamma': [0.5, 1, 2, 3, 4]}

# GridSearchCV
grid_search = GridSearchCV(SVC(kernel='rbf'), param_grid, cv=5, scoring='accuracy',n_jobs=-1)
```

```
grid_search.fit(X_train_scaled, y_train)
print("Best parameters found by GridSearchCV:")
print(grid_search.best_params_)


# Get the best model
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy of the best model: {accuracy*100:.2f}%")


mean_test_scores =
np.array(grid_search.cv_results_['mean_test_score']).reshape(len(param_grid['C']),
len(param_grid['gamma']))

# Plot a heatmap of accuracy for different combinations of C and gamma
plt.figure(figsize=(10, 6))
sns.heatmap(mean_test_scores, annot=True, fmt='.3f', cmap="viridis",
xticklabels=param_grid['gamma'], yticklabels=param_grid['C'])
plt.title('Accuracy vs C and Gamma Values')
plt.xlabel('Gamma')
plt.ylabel('C')
plt.show()


selector = SelectKBest(score_func=mutual_info_classif, k=10)
selector.fit(X_train_scaled, y_train)
selected_features = selector.get_support(indices=True)
selected_feature_names = X.columns[selected_features]

# Train SVM classifier with selected features
svm_selected = SVC(kernel='rbf', C=1, gamma=0.5, random_state=123)
svm_selected.fit(X_train_scaled[:, selected_features], y_train)

accuracy = svm_selected.score(X_test_scaled[:, selected_features], y_test)
print("Accuracy with selected features: {:.2f}%".format(accuracy * 100))

# Permutation importance
perm_importance = permutation_importance(svm_selected, X_test_scaled[:,
selected_features], y_test, n_repeats=30, random_state=42)

# Feature importances
importance_df = pd.DataFrame({'Feature': selected_feature_names, 'Importance':
perm_importance['importances_mean']})
importance_df.sort_values(by='Importance', ascending=False, inplace=True)
print(importance_df)
```

```
# Plotting
plt.figure(figsize=(10, 6))
sns.barplot(data=importance_df, x='Importance', y='Feature', palette='viridis')
plt.xlabel('Permutation Importance')
plt.ylabel('Feature')
plt.title('Feature Importance (Permutation)')
plt.show()


X_top = data[['AGE', 'BEDROOMS']]
y = data['OWNERSHP']
X_trainTop, X_testTop, y_trainTop, y_testTop =  train_test_split(X_top, y, test_size = 0.3,random_state = 123)
svm_classifier = SVC(kernel='rbf', C=1, gamma=0.5, cache_size=1000, verbose=True, max_iter=10000, random_state=123)
svm_classifier.fit(X_trainTop, y_trainTop)
y_pred = svm_classifier.predict(X_testTop)
accuracy = accuracy_score(y_testTop, y_pred)
print(f"Accuracy with top 2 predictors: {accuracy*100:.2f}%")


X_top = data[['BEDROOMS', 'AGE']]

y = data['OWNERSHP']
#spliting the data into train and test
X_trainTop, X_testTop, y_trainTop, y_testTop =  train_test_split(X_top, y, test_size = 0.3
                                    ,random_state = 123)

scaler = StandardScaler().fit(X_trainTop)
X_trainScaledTop = scaler.transform(X_trainTop)
X_testNewScaledTop = scaler.transform(X_testTop)
radsvc_top = SVC(kernel='rbf', C=1, gamma=0.5, cache_size=1000, verbose=True, max_iter=10000, random_state=123)
radsvc_top.fit(X_trainScaledTop, y_trainTop)


fig, ax = subplots(figsize=(8,8))
plot_svm(X_trainScaledTop,
        y_trainTop,
        radsvc_top,
        ax=ax)
ax.set_xlabel('BEDROOMS')
ax.set_ylabel('AGE')
plt.title('SVM Decision Boundary with Top 2 Predictor Variables')
```

```
plt.show()
```

Polynomial model

```
param_grid = {'C': [0.1, 1, 10, 100, 1000], 'degree': [2, 3, 4]}

grid_search = GridSearchCV(SVC(kernel='poly'), param_grid, cv=5, scoring='accuracy',n_jobs=-1)
grid_search.fit(X_train_scaled, y_train)

print("Best parameters found by GridSearchCV:")
print(grid_search.best_params_)


best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test_scaled)

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy of the best model: {accuracy*100:.2f}%")


mean_test_scores = np.array(grid_search.cv_results_['mean_test_score']).reshape(len(param_grid['C']), len(param_grid['degree']))

# Plot a heatmap of accuracy for different combinations of C and gamma
plt.figure(figsize=(10, 6))
sns.heatmap(mean_test_scores, annot=True, fmt='.3f', cmap="viridis", xticklabels=param_grid['degree'], yticklabels=param_grid['C'])
plt.title('Accuracy vs C and Degree Values')
plt.xlabel('Degree')
plt.ylabel('C')
plt.show()


selector = SelectKBest(score_func=mutual_info_classif, k=10)
selector.fit(X_train_scaled, y_train)
selected_features = selector.get_support(indices=True)
selected_feature_names = X.columns[selected_features]

# Train SVM classifier with selected features
```

```
svm_selected = SVC(kernel='poly', C=100, degree=3, random_state=123)
svm_selected.fit(X_train_scaled[:, selected_features], y_train)

# Evaluate the model on the testing data
accuracy = svm_selected.score(X_test_scaled[:, selected_features], y_test)
print("Accuracy with selected features: {:.2f}%".format(accuracy * 100))

# Permutation importance
perm_importance = permutation_importance(svm_selected, X_test_scaled[:,
selected_features], y_test, n_repeats=30, random_state=42)

# Feature importances
importance_df = pd.DataFrame({'Feature': selected_feature_names, 'Importance':
perm_importance['importances_mean']})
importance_df.sort_values(by='Importance', ascending=False, inplace=True)
print(importance_df)

plt.figure(figsize=(10, 6))
sns.barplot(data=importance_df, x='Importance', y='Feature', palette='viridis')
plt.xlabel('Permutation Importance')
plt.ylabel('Feature')
plt.title('Feature Importance (Permutation)')
plt.show()


X_top = data[['AGE', 'COSTWATR']]
y = data['OWNERSHP']
X_trainTop, X_testTop, y_trainTop, y_testTop = train_test_split(X_top, y, test_size =
0.3,random_state = 123)
svm_classifier = SVC(kernel='poly', C=100,degree=3, cache_size=1000, verbose=True,
max_iter=10000, random_state=123)
svm_classifier.fit(X_trainTop, y_trainTop)
y_pred = svm_classifier.predict(X_testTop)
accuracy = accuracy_score(y_testTop, y_pred)
print(f"Accuracy with top 2 predictors: {accuracy*100:.2f}%")


X_top = data[['AGE', 'COSTWATR']]

y = data['OWNERSHP']
#spliting the data into train and test
X_trainTop, X_testTop, y_trainTop, y_testTop =  train_test_split(X_top, y, test_size = 0.3
                                    ,random_state = 123)

scaler = StandardScaler().fit(X_trainTop)
X_trainScaledTop = scaler.transform(X_trainTop)
X_testNewScaledTop = scaler.transform(X_testTop)
```

```
polysvc_top = SVC(kernel='poly', C=100,degree=3, cache_size=1000, verbose=True, max_iter=10000, random_state=123)
polysvc_top.fit(X_trainScaledTop, y_trainTop)


fig, ax = subplots(figsize=(8,8))
plot_svm(X_trainScaledTop,
        y_trainTop,
        polysvc_top,
        ax=ax)
ax.set_xlabel('AGE')
ax.set_ylabel('COSTWATR')
plt.title('SVM Decision Boundary with Top 2 Predictor Variables')
 plt.show()
```