Housing Prices and Salary Prediction - SVR-Based Predictive Modeling

By Meenakshi Sharad Sethi

Disclaimer: This project is done on an individual capacity as part of academic learning and is not meant for commercial or professional use without further validation and refinement.

Introduction:

This project aims to develop predictive models for two crucial socioeconomic indicators: housing prices and salary. By leveraging machine learning techniques, specifically Support Vector Regression (SVR), we seek to analyze and predict these indicators based on various socioeconomic factors.

Project includes:

1. Boston Housing Data Analysis: Enhancing SVC Performance with 10-fold Cross-Validation

2. Salary Prediction Using SVR: Data Loading, Model Building, and Visualization

Boston Housing Data Analysis: Enhancing SVC Performance with 10-fold Cross-Validation

- (i) report average accuracy, confusion matrix, precision, recall, and F1 score; and
- (ii) use grid search to find the best C from C = [1, 5, 10, 50, 100, 500, 1000].

Loading necessary libraries

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score, GridSearchCV, train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recal
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.svm import SVR
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import GridSearchCV

import warnings
warnings.filterwarnings('ignore')
pd.set_option("display.max_columns", 60)
pd.set_option('display.max_rows', 50)
pd.set_option('display.width', 1000)
```

In [2]:
```
#### Loading the dataset

df = pd.read_csv("BostonHousing_full.csv")
df
```

Out[2]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	C
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	

506 rows × 14 columns

In [3]:

```
df.head()
```

Out[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	CAT
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	

In [4]:

```
df.tail()
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	CA
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
```

```
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64  
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    int64  
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  CATMEDV  506 non-null    object 
dtypes: float64(10), int64(3), object(1)
memory usage: 55.5+ KB
```

In [6]:

```
# Statitical summary for numerical features
df.describe()
```

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DI
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.79504
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.10571
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.12960
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.10017
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.20745
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.18842
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.12650

In [7]:

```
# Display unique values for all columns in the dataset
cols = df.columns

# for each column
for col in cols:
    print(col)

    # get a list of unique values
    unique = df[col].unique()
    print(unique, '\n====================================\n\n')
```

```
CRIM
[6.32000e-03 2.73100e-02 2.72900e-02 3.23700e-02 6.90500e-02 2.98500e-02
 8.82900e-02 1.44550e-01 2.11240e-01 1.70040e-01 2.24890e-01 1.17470e-01
 9.37800e-02 6.29760e-01 6.37960e-01 6.27390e-01 1.05393e+00 7.84200e-01
 8.02710e-01 7.25800e-01 1.25179e+00 8.52040e-01 1.23247e+00 9.88430e-01
 7.50260e-01 8.40540e-01 6.71910e-01 9.55770e-01 7.72990e-01 1.00245e+00
 1.13081e+00 1.35472e+00 1.38799e+00 1.15172e+00 1.61282e+00 6.41700e-02
 9.74400e-02 8.01400e-02 1.75050e-01 2.76300e-02 3.35900e-02 1.27440e-01
 1.41500e-01 1.59360e-01 1.22690e-01 1.71420e-01 1.88360e-01 2.29270e-01
 2.53870e-01 2.19770e-01 8.87300e-02 4.33700e-02 5.36000e-02 4.98100e-02
 1.36000e-02 1.31100e-02 2.05500e-02 1.43200e-02 1.54450e-01 1.03280e-01
```

1.49320e-01 1.71710e-01 1.10270e-01 1.26500e-01 1.95100e-02 3.58400e-02
4.37900e-02 5.78900e-02 1.35540e-01 1.28160e-01 8.82600e-02 1.58760e-01
9.16400e-02 1.95390e-01 7.89600e-02 9.51200e-02 1.01530e-01 8.70700e-02
5.64600e-02 8.38700e-02 4.11300e-02 4.46200e-02 3.65900e-02 3.55100e-02
5.05900e-02 5.73500e-02 5.18800e-02 7.15100e-02 5.66000e-02 5.30200e-02
4.68400e-02 3.93200e-02 4.20300e-02 2.87500e-02 4.29400e-02 1.22040e-01
1.15040e-01 1.20830e-01 8.18700e-02 6.86000e-02 1.48660e-01 1.14320e-01
2.28760e-01 2.11610e-01 1.39600e-01 1.32620e-01 1.71200e-01 1.31170e-01
1.28020e-01 2.63630e-01 1.07930e-01 1.00840e-01 1.23290e-01 2.22120e-01
1.42310e-01 1.71340e-01 1.31580e-01 1.50980e-01 1.30580e-01 1.44760e-01
6.89900e-02 7.16500e-02 9.29900e-02 1.50380e-01 9.84900e-02 1.69020e-01
3.87350e-01 2.59150e-01 3.25430e-01 8.81250e-01 3.40060e-01 1.19294e+00
5.90050e-01 3.29820e-01 9.76170e-01 5.57780e-01 3.22640e-01 3.52330e-01
2.49800e-01 5.44520e-01 2.90900e-01 1.62864e+00 3.32105e+00 4.09740e+00
2.77974e+00 2.37934e+00 2.15505e+00 2.36862e+00 2.33099e+00 2.73397e+00
1.65660e+00 1.49632e+00 1.12658e+00 2.14918e+00 1.41385e+00 3.53501e+00
2.44668e+00 1.22358e+00 1.34284e+00 1.42502e+00 1.27346e+00 1.46336e+00
1.83377e+00 1.51902e+00 2.24236e+00 2.92400e+00 2.01019e+00 1.80028e+00
2.30040e+00 2.44953e+00 1.20742e+00 2.31390e+00 1.39140e-01 9.17800e-02
8.44700e-02 6.66400e-02 7.02200e-02 5.42500e-02 6.64200e-02 5.78000e-02
6.58800e-02 6.88800e-02 9.10300e-02 1.00080e-01 8.30800e-02 6.04700e-02
5.60200e-02 7.87500e-02 1.25790e-01 8.37000e-02 9.06800e-02 6.91100e-02
8.66400e-02 2.18700e-02 1.43900e-02 1.38100e-02 4.01100e-02 4.66600e-02
3.76800e-02 3.15000e-02 1.77800e-02 3.44500e-02 2.17700e-02 3.51000e-02
2.00900e-02 1.36420e-01 2.29690e-01 2.51990e-01 1.35870e-01 4.35710e-01
1.74460e-01 3.75780e-01 2.17190e-01 1.40520e-01 2.89550e-01 1.98020e-01
4.56000e-02 7.01300e-02 1.10690e-01 1.14250e-01 3.58090e-01 4.07710e-01
6.23560e-01 6.14700e-01 3.15330e-01 5.26930e-01 3.82140e-01 4.12380e-01
2.98190e-01 4.41780e-01 5.37000e-01 4.62960e-01 5.75290e-01 3.31470e-01
4.47910e-01 3.30450e-01 5.20580e-01 5.11830e-01 8.24400e-02 9.25200e-02
1.13290e-01 1.06120e-01 1.02900e-01 1.27570e-01 2.06080e-01 1.91330e-01
3.39830e-01 1.96570e-01 1.64390e-01 1.90730e-01 1.40300e-01 2.14090e-01
8.22100e-02 3.68940e-01 4.81900e-02 3.54800e-02 1.53800e-02 6.11540e-01
6.63510e-01 6.56650e-01 5.40110e-01 5.34120e-01 5.20140e-01 8.25260e-01
5.50070e-01 7.61620e-01 7.85700e-01 5.78340e-01 5.40500e-01 9.06500e-02
2.99160e-01 1.62110e-01 1.14600e-01 2.21880e-01 5.64400e-02 9.60400e-02
1.04690e-01 6.12700e-02 7.97800e-02 2.10380e-01 3.57800e-02 3.70500e-02
6.12900e-02 1.50100e-02 9.06000e-03 1.09600e-02 1.96500e-02 3.87100e-02
4.59000e-02 4.29700e-02 3.50200e-02 7.88600e-02 3.61500e-02 8.26500e-02
8.19900e-02 1.29320e-01 5.37200e-02 1.41030e-01 6.46600e-02 5.56100e-02
4.41700e-02 3.53700e-02 9.26600e-02 1.00000e-01 5.51500e-02 5.47900e-02
7.50300e-02 4.93200e-02 4.92980e-01 3.49400e-01 2.63548e+00 7.90410e-01
2.61690e-01 2.69380e-01 3.69200e-01 2.53560e-01 3.18270e-01 2.45220e-01
4.02020e-01 4.75470e-01 1.67600e-01 1.81590e-01 3.51140e-01 2.83920e-01
3.41090e-01 1.91860e-01 3.03470e-01 2.41030e-01 6.61700e-02 6.72400e-02
4.54400e-02 5.02300e-02 3.46600e-02 5.08300e-02 3.73800e-02 3.96100e-02
3.42700e-02 3.04100e-02 3.30600e-02 5.49700e-02 6.15100e-02 1.30100e-02
2.49800e-02 2.54300e-02 3.04900e-02 3.11300e-02 6.16200e-02 1.87000e-02
2.89900e-02 6.21100e-02 7.95000e-02 7.24400e-02 1.70900e-02 4.30100e-02
1.06590e-01 8.98296e+00 3.84970e+00 5.20177e+00 4.26131e+00 4.54192e+00
3.83684e+00 3.67822e+00 4.22239e+00 3.47428e+00 4.55587e+00 3.69695e+00
1.35222e+01 4.89822e+00 5.66998e+00 6.53876e+00 9.23230e+00 8.26725e+00
1.11081e+01 1.84982e+01 1.96091e+01 1.52880e+01 9.82349e+00 2.36482e+01
1.78667e+01 8.89762e+01 1.58744e+01 9.18702e+00 7.99248e+00 2.00849e+01
1.68118e+01 2.43938e+01 2.25971e+01 1.43337e+01 8.15174e+00 6.96215e+00
5.29305e+00 1.15779e+01 8.64476e+00 1.33598e+01 8.71675e+00 5.87205e+00
7.67202e+00 3.83518e+01 9.91655e+00 2.50461e+01 1.42362e+01 9.59571e+00
2.48017e+01 4.15292e+01 6.79208e+01 2.07162e+01 1.19511e+01 7.40389e+00
1.44383e+01 5.11358e+01 1.40507e+01 1.88110e+01 2.86558e+01 4.57461e+01
1.80846e+01 1.08342e+01 2.59406e+01 7.35341e+01 1.18123e+01 1.10874e+01
7.02259e+00 1.20482e+01 7.05042e+00 8.79212e+00 1.58603e+01 1.22472e+01
3.76619e+01 7.36711e+00 9.33889e+00 8.49213e+00 1.00623e+01 6.44405e+00
5.58107e+00 1.39134e+01 1.11604e+01 1.44208e+01 1.51772e+01 1.36781e+01
9.39063e+00 2.20511e+01 9.72418e+00 5.66637e+00 9.96654e+00 1.28023e+01

```
 1.06718e+01 6.28807e+00 9.92485e+00 9.32909e+00 7.52601e+00 6.71772e+00
 5.44114e+00 5.09017e+00 8.24809e+00 9.51363e+00 4.75237e+00 4.66883e+00
 8.20058e+00 7.75223e+00 6.80117e+00 4.81213e+00 3.69311e+00 6.65492e+00
 5.82115e+00 7.83932e+00 3.16360e+00 3.77498e+00 4.42228e+00 1.55757e+01
 1.30751e+01 4.34879e+00 4.03841e+00 3.56868e+00 4.64689e+00 8.05579e+00
 6.39312e+00 4.87141e+00 1.50234e+01 1.02330e+01 5.82401e+00 5.70818e+00
 5.73116e+00 2.81838e+00 2.37857e+00 3.67367e+00 5.69175e+00 4.83567e+00
 1.50860e-01 1.83370e-01 2.07460e-01 1.05740e-01 1.11320e-01 1.73310e-01
 2.79570e-01 1.78990e-01 2.89600e-01 2.68380e-01 2.39120e-01 1.77830e-01
 2.24380e-01 6.26300e-02 4.52700e-02 6.07600e-02 1.09590e-01 4.74100e-02]
====================================


ZN
[ 18.    0.   12.5  75.   21.   90.   85.  100.   25.   17.5  80.   28.
  45.   60.   95.   82.5  30.   22.   20.   40.   55.   52.5  70.   34.
  33.   35. ]
====================================


INDUS
[ 2.31  7.07  2.18  7.87  8.14  5.96  2.95  6.91  5.64  4.    1.22  0.74
  1.32  5.13  1.38  3.37  6.07 10.81 12.83  4.86  4.49  3.41 15.04  2.89
  8.56 10.01 25.65 21.89 19.58  4.05  2.46  3.44  2.93  0.46  1.52  1.47
  2.03  2.68 10.59 13.89  6.2   4.93  5.86  3.64  3.75  3.97  6.96  6.41
  3.33  1.21  2.97  2.25  1.76  5.32  4.95 13.92  2.24  6.09  9.9   7.38
  3.24  6.06  5.19  1.89  3.78  4.39  4.15  2.01  1.25  1.69  2.02  1.91
 18.1  27.74  9.69 11.93]
====================================


CHAS
[0 1]
====================================


NOX
[0.538  0.469  0.458  0.524  0.499  0.428  0.448  0.439  0.41   0.403
 0.411  0.453  0.4161 0.398  0.409  0.413  0.437  0.426  0.449  0.489
 0.464  0.445  0.52   0.547  0.581  0.624  0.871  0.605  0.51   0.488
 0.401  0.422  0.404  0.415  0.55   0.507  0.504  0.431  0.392  0.394
 0.647  0.575  0.447  0.4429 0.4    0.389  0.385  0.405  0.433  0.472
 0.544  0.493  0.46   0.4379 0.515  0.442  0.518  0.484  0.429  0.435
 0.77   0.718  0.631  0.668  0.671  0.7    0.693  0.659  0.597  0.679
 0.614  0.584  0.713  0.74   0.655  0.58   0.532  0.583  0.609  0.585
 0.573 ]
====================================


RM
[6.575 6.421 7.185 6.998 7.147 6.43  6.012 6.172 5.631 6.004 6.377 6.009
 5.889 5.949 6.096 5.834 5.935 5.99  5.456 5.727 5.57  5.965 6.142 5.813
 5.924 5.599 6.047 6.495 6.674 5.713 6.072 5.95  5.701 5.933 5.841 5.85
 5.966 6.595 7.024 6.77  6.169 6.211 6.069 5.682 5.786 6.03  5.399 5.602
 5.963 6.115 6.511 5.998 5.888 7.249 6.383 6.816 6.145 5.927 5.741 6.456
 6.762 7.104 6.29  5.787 5.878 5.594 5.885 6.417 5.961 6.065 6.245 6.273
 6.286 6.279 6.14  6.232 5.874 6.727 6.619 6.302 6.167 6.389 6.63  6.015
 6.121 7.007 7.079 6.405 6.442 6.249 6.625 6.163 8.069 7.82  7.416 6.781
 6.137 5.851 5.836 6.127 6.474 6.229 6.195 6.715 5.913 6.092 6.254 5.928
 6.176 6.021 5.872 5.731 5.87  5.856 5.879 5.986 5.613 5.693 6.431 5.637
 6.458 6.326 6.372 5.822 5.757 6.335 5.942 6.454 5.857 6.151 6.174 5.019
 5.403 5.468 4.903 6.13  5.628 4.926 5.186 5.597 6.122 5.404 5.012 5.709
 6.129 6.152 5.272 6.943 6.066 6.51  6.25  7.489 7.802 8.375 5.854 6.101
 7.929 5.877 6.319 6.402 5.875 5.88  5.572 6.416 5.859 6.546 6.02  6.315
```

```
 6.86  6.98  7.765 6.144 7.155 6.563 5.604 6.153 7.831 6.782 6.556 6.951
 6.739 7.178 6.8   6.604 7.875 7.287 7.107 7.274 6.975 7.135 6.162 7.61
 7.853 8.034 5.891 5.783 6.064 5.344 5.96  5.807 6.375 5.412 6.182 6.642
 5.951 6.373 6.164 6.879 6.618 8.266 8.725 8.04  7.163 7.686 6.552 5.981
 7.412 8.337 8.247 6.726 6.086 6.631 7.358 6.481 6.606 6.897 6.095 6.358
 6.393 5.593 5.605 6.108 6.226 6.433 6.718 6.487 6.438 6.957 8.259 5.876
 7.454 8.704 7.333 6.842 7.203 7.52  8.398 7.327 7.206 5.56  7.014 8.297
 7.47  5.92  6.24  6.538 7.691 6.758 6.854 7.267 6.826 6.482 6.812 6.968
 7.645 7.923 7.088 6.453 6.23  6.209 6.565 6.861 7.148 6.678 6.549 5.79
 6.345 7.041 6.871 6.59  6.982 7.236 6.616 7.42  6.849 6.635 5.972 4.973
 6.023 6.266 6.567 5.705 5.914 5.782 6.382 6.113 6.426 6.376 6.041 5.708
 6.415 6.312 6.083 5.868 6.333 5.706 6.031 6.316 6.31  6.037 5.869 5.895
 6.059 5.985 5.968 7.241 6.54  6.696 6.874 6.014 5.898 6.516 6.939 6.49
 6.579 5.884 6.728 5.663 5.936 6.212 6.395 6.112 6.398 6.251 5.362 5.803
 8.78  3.561 4.963 3.863 4.97  6.683 7.016 6.216 4.906 4.138 7.313 6.649
 6.794 6.38  6.223 6.545 5.536 5.52  4.368 5.277 4.652 5.    4.88  5.39
 6.051 5.036 6.193 5.887 6.471 5.747 5.453 5.852 5.987 6.343 6.404 5.349
 5.531 5.683 5.608 5.617 6.852 6.657 4.628 5.155 4.519 6.434 5.304 5.957
 6.824 6.411 6.006 5.648 6.103 5.565 5.896 5.837 6.202 6.348 6.833 6.425
 6.436 6.208 6.629 6.461 5.627 5.818 6.406 6.219 6.485 6.459 6.341 6.185
 6.749 6.655 6.297 7.393 6.525 5.976 6.301 6.081 6.701 6.317 6.513 5.759
 5.952 6.003 5.926 6.437 5.427 6.484 6.242 6.75  7.061 5.762 5.871 6.114
 5.905 5.454 5.414 5.093 5.983 5.707 5.67  5.794 6.019 5.569 6.027 6.593
 6.12  6.976]
====================================


AGE
[ 65.2  78.9  61.1  45.8  54.2  58.7  66.6  96.1 100.   85.9  94.3  82.9
  39.   61.8  84.5  56.5  29.3  81.7  36.6  69.5  98.1  89.2  91.7  94.1
  85.7  90.3  88.8  94.4  87.3  82.   95.   96.9  68.2  61.4  41.5  30.2
  21.8  15.8   2.9   6.6   6.5  40.   33.8  33.3  85.5  95.3  62.   45.7
  63.   21.1  21.4  47.6  21.9  35.7  40.5  29.2  47.2  66.2  93.4  67.8
  43.4  59.5  17.8  31.1  36.8  33.   17.5   7.8   6.2   6.   45.   74.5
  53.7  33.5  70.4  32.2  46.7  48.   56.1  45.1  56.8  86.3  63.1  66.1
  73.9  53.6  28.9  77.3  57.8  69.6  76.   36.9  62.5  79.9  71.3  85.4
  87.4  90.   96.7  91.9  85.2  97.1  91.2  54.4  81.6  92.9  95.4  84.2
  88.2  72.5  82.6  73.1  69.7  84.1  97.   95.8  88.4  95.6  96.   98.8
  94.7  98.9  97.7  97.9  98.4  98.2  93.5  93.6  97.8  95.7  93.8  94.9
  97.3  88.   98.5  94.   97.4  92.6  90.8  93.9  91.8  93.   96.2  79.2
  95.2  94.6  88.5  68.7  33.1  73.4  74.4  58.4  83.3  62.2  92.2  89.8
  68.8  41.1  29.1  38.9  21.5  30.8  26.3   9.9  18.8  32.   34.1  38.3
  15.3  13.9  38.4  15.7  33.2  31.9  22.3  52.5  72.7  59.1  92.1  88.6
  53.8  32.3   9.8  42.4  56.   85.1  92.4  91.3  77.7  80.8  78.3  83.
  86.5  17.   68.1  76.9  73.3  66.5  61.5  76.5  71.6  18.5  42.2  54.3
  65.1  52.9  70.2  34.9  49.1  13.    8.9   6.8   8.4  19.1  34.2  86.9
  81.8  89.4  91.5  94.5  91.6  62.8  84.6  67.   52.6  42.1  16.3  51.8
  32.9  42.8  49.   27.6  32.1  64.5  37.2  49.7  24.8  20.8  31.5  31.3
  45.6  22.9  27.9  27.7  23.4  18.4  42.3  51.   58.   20.1  10.   47.4
  40.4  17.7  58.1  71.9  70.3  82.5  76.7  37.8  52.8  90.4  82.8  83.2
  71.7  67.2  58.8  52.3  49.9  74.3  40.1  14.7  43.7  25.8  17.2  28.4
  23.3  38.1  38.5  34.5  46.3  59.6  37.3  45.4  58.5  49.3  59.7  56.4
  28.1  48.5  29.7  44.4  35.9  36.1  19.5  91.   83.4  81.3  91.1  89.
  87.9  91.4  96.8  97.5  89.6  93.3  99.1  89.5  77.8  89.1  87.6  70.6
  78.7  78.1  86.1  74.8  97.2  96.6  94.8  96.4  98.7  98.3  99.3  80.3
  83.7  84.4  89.9  65.4  48.2  84.7  71.   56.7  84.   90.7  75.   67.6
  64.7  74.9  77.   40.3  41.9  51.9  79.8  53.2  92.7  98.   83.5  54.
  42.6  28.8  72.9  65.3  73.5  79.7  69.1  89.3]
====================================


DIS
[ 4.09    4.9671  6.0622  5.5605  5.9505  6.0821  6.5921  6.3467  6.2267
  5.4509  4.7075  4.4619  4.4986  4.2579  3.7965  3.7979  4.0123  3.9769
```

```
  4.0952  4.3996  4.4546  4.682   4.4534  4.4547  4.239   4.233   4.175
  3.99    3.7872  3.7598  3.3603  3.3779  3.9342  3.8473  5.4011  5.7209
  5.1004  5.6894  5.87    6.0877  6.8147  7.3197  8.6966  9.1876  8.3248
  7.8148  6.932   7.2254  6.8185  7.2255  7.9809  9.2229  6.6115  6.498
  5.2873  4.2515  4.5026  4.0522  4.0905  5.0141  5.4007  4.7794  4.4377
  4.4272  3.7476  3.4217  3.4145  3.0923  3.0921  3.6659  3.615   3.4952
  2.7778  2.8561  2.7147  2.421   2.1069  2.211   2.1224  2.4329  2.5451
  2.6775  2.3534  2.548   2.2565  2.4631  2.7301  2.7474  2.4775  2.7592
  2.2577  2.1974  2.0869  1.9444  2.0063  1.9929  1.7572  1.7883  1.8125
  1.9799  2.1185  2.271   2.3274  2.4699  2.346   2.1107  1.9669  1.8498
  1.6686  1.6687  1.6119  1.4394  1.3216  1.4118  1.3459  1.4191  1.5166
  1.4608  1.5296  1.5257  1.618   1.5916  1.6102  1.6232  1.7494  1.7455
  1.7364  1.8773  1.7573  1.7659  1.7984  1.9709  2.0407  2.162   2.422
  2.2834  2.0459  2.4259  2.1     2.2625  2.3887  2.5961  2.6463  2.7019
  3.1323  3.5549  3.3175  2.9153  2.829   2.741   2.5979  2.7006  2.847
  2.9879  3.2797  3.1992  3.7886  4.5667  6.4798  6.2196  5.6484  7.309
  7.6534  6.27    5.118   3.9454  4.3549  4.2392  3.875   3.8771  3.665
  3.6526  3.5875  3.1121  3.4211  2.8893  3.3633  2.8617  3.048   3.2721
  2.8944  3.2157  3.3751  3.6715  3.8384  3.6519  4.148   6.1899  6.3361
  7.0355  7.9549  8.0555  7.8265  7.3967  8.9067  9.2203  1.801   1.8946
  2.0107  2.1121  2.1398  2.2885  2.0788  1.9301  1.9865  2.1329  2.4216
  2.872   3.9175  4.429   4.3665  4.0776  4.2673  4.7872  4.8628  4.1403
  4.1007  4.6947  5.2447  5.2119  5.885   7.3073  9.0892  7.3172  5.1167
  5.5027  5.9604  6.32    7.8278  5.4917  4.022   3.37    3.0992  3.1827
  3.1025  2.5194  2.6403  2.834   3.2628  3.6023  3.945   3.9986  4.0317
  3.5325  4.0019  4.5404  4.7211  5.4159  5.2146  5.8736  6.6407  6.4584
  5.9853  5.2311  5.615   4.8122  7.0379  6.2669  5.7321  6.4654  8.0136
  8.5353  8.344   8.7921 10.7103 12.1265 10.5857  2.1222  2.5052  2.7227
  2.5091  2.5182  2.2955  2.1036  1.9047  1.6132  1.7523  1.5106  1.3325
  1.3567  1.2024  1.1691  1.1296  1.1742  1.137   1.3163  1.3449  1.358
  1.3861  1.4165  1.5192  1.5804  1.5331  1.4395  1.4261  1.4672  1.5184
  1.5895  1.7281  1.9265  2.1678  1.77    1.7912  1.7821  1.7257  1.6768
  1.6334  1.4896  1.5004  1.5888  1.5741  1.639   1.7028  1.6074  1.4254
  1.1781  1.2852  1.4547  1.4655  1.413   1.5275  1.5539  1.5894  1.6582
  1.8347  1.8195  1.6475  1.8026  1.794   1.8589  1.8746  1.9512  2.0218
  2.0635  1.9096  1.9976  1.8629  1.9356  1.9682  2.0527  2.0882  2.2004
  2.3158  2.2222  2.1247  2.0026  1.9142  1.8206  1.8172  1.8662  2.0651
  2.0048  1.9784  1.8956  1.9879  2.072   2.198   2.2616  2.185   2.3236
  2.3552  2.3682  2.4527  2.4961  2.4358  2.5806  2.7792  2.7831  2.7175
  2.5975  2.5671  2.7344  2.8016  2.9634  3.0665  2.8715  2.5403  2.9084
  2.8237  3.0334  3.0993  2.8965  2.5329  2.4298  2.206   2.3053  2.1007
  2.1705  3.4242  3.3317  3.4106  4.0983  3.724   3.9917  3.5459  3.1523
  1.8209  1.7554  1.8226  1.8681  2.1099  2.3817  2.7986  2.8927  2.4091
  2.3999  2.4982  2.4786  2.2875  2.1675  2.3889  2.505 ]
====================================


RAD
[ 1  2  3  5  4  8  6  7 24]
====================================


TAX
[296 242 222 311 307 279 252 233 243 469 226 313 256 284 216 337 345 305
 398 281 247 270 276 384 432 188 437 403 193 265 255 329 402 348 224 277
 300 330 315 244 264 223 254 198 285 241 293 245 289 358 304 287 430 422
 370 352 351 280 335 411 187 334 666 711 391 273]
====================================


PTRATIO
[15.3 17.8 18.7 15.2 21.  19.2 18.3 17.9 16.8 21.1 17.3 15.1 19.7 18.6
 16.1 18.9 19.  18.5 18.2 18.  20.9 19.1 21.2 14.7 16.6 15.6 14.4 12.6
 17.  16.4 17.4 15.9 13.  17.6 14.9 13.6 16.  14.8 18.4 19.6 16.9 20.2
```

```
 15.5 18.8 22.  20.1]
====================================


B
[3.9690e+02 3.9283e+02 3.9463e+02 3.9412e+02 3.9560e+02 3.8663e+02
 3.8671e+02 3.9252e+02 3.9050e+02 3.8002e+02 3.9562e+02 3.8685e+02
 3.8675e+02 2.8899e+02 3.9095e+02 3.7657e+02 3.9253e+02 3.9454e+02
 3.9433e+02 3.0342e+02 3.7688e+02 3.0638e+02 3.8794e+02 3.8023e+02
 3.6017e+02 3.7673e+02 2.3260e+02 3.5877e+02 2.4831e+02 3.7756e+02
 3.9343e+02 3.9563e+02 3.8541e+02 3.8337e+02 3.9446e+02 3.8939e+02
 3.9274e+02 3.9556e+02 3.9397e+02 3.9593e+02 3.9290e+02 3.9068e+02
 3.9511e+02 3.7808e+02 3.9558e+02 3.9324e+02 3.9621e+02 3.8373e+02
 3.7694e+02 3.9091e+02 3.7717e+02 3.9492e+02 3.8323e+02 3.7366e+02
 3.8696e+02 3.8640e+02 3.9606e+02 3.9064e+02 3.9230e+02 3.9599e+02
 3.9515e+02 3.9218e+02 3.9355e+02 3.9501e+02 3.9633e+02 3.5798e+02
 3.9183e+02 3.9353e+02 3.9476e+02 7.0800e+01 3.9447e+02 3.9269e+02
 3.9405e+02 3.9567e+02 3.8769e+02 3.9524e+02 3.9123e+02 3.9349e+02
 3.9559e+02 3.9495e+02 3.8874e+02 3.4491e+02 3.9330e+02 3.9451e+02
 3.3863e+02 3.9150e+02 3.8915e+02 3.7767e+02 3.7809e+02 3.7031e+02
 3.7938e+02 3.8502e+02 3.5929e+02 3.9211e+02 3.9504e+02 3.8576e+02
 3.8869e+02 2.6276e+02 3.9467e+02 3.7825e+02 3.9408e+02 3.9204e+02
 3.8808e+02 1.7291e+02 1.6927e+02 3.9171e+02 3.5699e+02 3.5185e+02
 3.7280e+02 3.4160e+02 3.4328e+02 2.6195e+02 3.2102e+02 8.8010e+01
 8.8630e+01 3.6343e+02 3.5389e+02 3.6431e+02 3.3892e+02 3.7443e+02
 3.8961e+02 3.8845e+02 2.4016e+02 3.6930e+02 2.2761e+02 2.9709e+02
 3.3004e+02 2.9229e+02 3.4813e+02 3.9550e+02 3.9323e+02 3.9096e+02
 3.9127e+02 3.9100e+02 3.8711e+02 3.9263e+02 3.9387e+02 3.8284e+02
 3.7768e+02 3.8971e+02 3.9049e+02 3.9337e+02 3.7670e+02 3.9423e+02
 3.5431e+02 3.9220e+02 3.8430e+02 3.9377e+02 3.9538e+02 3.9278e+02
 3.9055e+02 3.9487e+02 3.8943e+02 3.8132e+02 3.9325e+02 3.9094e+02
 3.8581e+02 3.4893e+02 3.9363e+02 3.9280e+02 3.9374e+02 3.9170e+02
 3.9039e+02 3.8505e+02 3.8200e+02 3.8738e+02 3.7208e+02 3.7751e+02
 3.8034e+02 3.7835e+02 3.7614e+02 3.8591e+02 3.7895e+02 3.6020e+02
 3.7675e+02 3.9007e+02 3.7941e+02 3.8378e+02 3.9125e+02 3.9462e+02
 3.7275e+02 3.7471e+02 3.7249e+02 3.8913e+02 3.9018e+02 3.9628e+02
 3.7707e+02 3.8609e+02 3.9289e+02 3.9518e+02 3.8634e+02 3.8970e+02
 3.8329e+02 3.9193e+02 3.8837e+02 3.8686e+02 3.9342e+02 3.8789e+02
 3.9240e+02 3.8407e+02 3.8454e+02 3.9030e+02 3.9134e+02 3.8865e+02
 3.9496e+02 3.9077e+02 3.8925e+02 3.9345e+02 3.8731e+02 3.9223e+02
 3.9552e+02 3.9472e+02 3.7172e+02 3.9285e+02 3.6824e+02 3.7158e+02
 3.9086e+02 3.9575e+02 3.8361e+02 3.9043e+02 3.9368e+02 3.9336e+02
 3.9624e+02 3.5045e+02 3.9630e+02 3.9339e+02 3.9569e+02 3.9642e+02
 3.9070e+02 3.9521e+02 3.9623e+02 3.9113e+02 3.8244e+02 3.7521e+02
 3.6857e+02 3.9402e+02 3.6225e+02 3.8940e+02 3.9481e+02 3.9614e+02
 3.9474e+02 3.8996e+02 3.8797e+02 3.8564e+02 3.6461e+02 3.9243e+02
 3.8985e+02 3.7078e+02 3.9233e+02 3.8446e+02 3.8280e+02 3.7604e+02
 3.7773e+02 3.9543e+02 3.9074e+02 3.7456e+02 3.5065e+02 3.8079e+02
 3.5304e+02 3.5455e+02 3.5470e+02 3.1603e+02 1.3142e+02 3.7552e+02
 3.7533e+02 3.9205e+02 3.6615e+02 3.4788e+02 3.6302e+02 2.8583e+02
 3.7292e+02 3.9443e+02 3.7838e+02 3.9198e+02 3.9310e+02 3.3816e+02
 3.7611e+02 3.2946e+02 3.8497e+02 3.7022e+02 3.3209e+02 3.1464e+02
 1.7936e+02 2.6000e+00 3.5050e+01 2.8790e+01 2.1097e+02 8.8270e+01
 2.7250e+01 2.1570e+01 1.2736e+02 1.6450e+01 4.8450e+01 3.1875e+02
 3.1998e+02 2.9155e+02 2.5200e+00 3.6500e+00 7.6800e+00 2.4650e+01
 1.8820e+01 9.6730e+01 6.0720e+01 8.3450e+01 8.1330e+01 9.7950e+01
 1.0019e+02 1.0063e+02 1.0985e+02 2.7490e+01 9.3200e+00 6.8950e+01
 3.9145e+02 3.8596e+02 3.8673e+02 2.4052e+02 4.3060e+01 3.1801e+02
 3.8852e+02 3.0421e+02 3.2000e-01 3.5529e+02 3.8509e+02 3.7587e+02
 6.6800e+00 5.0920e+01 1.0480e+01 3.5000e+00 2.7221e+02 2.5523e+02
 3.9143e+02 3.9382e+02 3.3440e+02 2.2010e+01 3.3129e+02 3.6874e+02
 3.9533e+02 3.7468e+02 3.5258e+02 3.0276e+02 3.4948e+02 3.7970e+02
 3.8332e+02 3.9307e+02 3.9528e+02 3.9292e+02 3.7073e+02 3.8862e+02
 3.9268e+02 3.8822e+02 3.9509e+02 3.4405e+02 3.1843e+02 3.9011e+02
```

```
 3.9329e+02 3.9577e+02 3.9199e+02]
====================================


LSTAT
[ 4.98  9.14  4.03  2.94  5.33  5.21 12.43 19.15 29.93 17.1  20.45 13.27
 15.71  8.26 10.26  8.47  6.58 14.67 11.69 11.28 21.02 13.83 18.72 19.88
 16.3  16.51 14.81 17.28 12.8  11.98 22.6  13.04 27.71 18.35 20.34  9.68
 11.41  8.77 10.13  4.32  1.98  4.84  5.81  7.44  9.55 10.21 14.15 18.8
 30.81 16.2  13.45  9.43  5.28  8.43 14.8   4.81  5.77  3.95  6.86  9.22
 13.15 14.44  6.73  9.5   8.05  4.67 10.24  8.1  13.09  8.79  6.72  9.88
  5.52  7.54  6.78  8.94 11.97 10.27 12.34  9.1   5.29  7.22  7.51  9.62
  6.53 12.86  8.44  5.5   5.7   8.81  8.2   8.16  6.21 10.59  6.65 11.34
  4.21  3.57  6.19  9.42  7.67 10.63 13.44 12.33 16.47 18.66 14.09 12.27
 15.55 13.   10.16 16.21 17.09 10.45 15.76 12.04 10.3  15.37 13.61 14.37
 14.27 17.93 25.41 17.58 27.26 17.19 15.39 18.34 12.6  12.26 11.12 15.03
 17.31 16.96 16.9  14.59 21.32 18.46 24.16 34.41 26.82 26.42 29.29 27.8
 16.65 29.53 28.32 21.45 14.1  13.28 12.12 15.79 15.12 15.02 16.14  4.59
  6.43  7.39  1.73  1.92  3.32 11.64  9.81  3.7  12.14 11.1  11.32 14.43
 12.03 14.69  9.04  9.64 10.11  6.29  6.92  5.04  7.56  9.45  4.82  5.68
 13.98  4.45  6.68  4.56  5.39  5.1   4.69  2.87  5.03  4.38  2.97  4.08
  8.61  6.62  7.43  3.11  3.81  2.88 10.87 10.97 18.06 14.66 23.09 17.27
 23.98 16.03  9.38 29.55  9.47 13.51  9.69 17.92 10.5   9.71 21.46  9.93
  7.6   4.14  4.63  3.13  6.36  3.92  3.76 11.65  5.25  2.47 10.88  9.54
  4.73  7.37 11.38 12.4  11.22  5.19 12.5   9.16 10.15  9.52  6.56  5.9
  3.59  3.53  3.54  6.57  9.25  5.12  7.79  6.9   9.59  7.26  5.91 11.25
 14.79  3.16 13.65  6.59  7.73  2.98  6.05  4.16  7.19  4.85  3.01  7.85
  8.23 12.93  7.14  9.51  3.33  3.56  4.7   8.58 10.4   6.27 15.84  4.97
  4.74  6.07  8.67  4.86  6.93  8.93  6.47  7.53  4.54  9.97 12.64  5.98
 11.72  7.9   9.28 11.5  18.33 15.94 10.36 12.73  7.2   6.87  7.7  11.74
  6.12  5.08  6.15 12.79  7.34  9.09  7.83  6.75  8.01  9.8  10.56  8.51
  9.74  9.29  5.49  8.65  7.18  4.61 10.53 12.67  5.99  5.89  4.5   5.57
 17.6  11.48 14.19 10.19 14.64  7.12 14.   13.33  3.26  3.73  2.96  9.53
  8.88 34.77 37.97 23.24 21.24 23.69 21.78 17.21 21.08 23.6  24.56 30.63
 28.28 31.99 30.62 20.85 17.11 18.76 25.68 15.17 16.35 17.12 19.37 19.92
 30.59 29.97 26.77 20.32 20.31 19.77 27.38 22.98 23.34 12.13 26.4  19.78
 21.22 34.37 20.08 36.98 29.05 25.79 26.64 20.62 22.74 15.7  23.29 17.16
 24.39 15.69 14.52 21.52 24.08 17.64 19.69 16.22 23.27 18.05 26.45 34.02
 22.88 22.11 19.52 16.59 18.85 23.79 17.79 16.44 18.13 19.31 17.44 17.73
 16.74 18.71 19.01 16.94 16.23 14.7  16.42 14.65 13.99 10.29 13.22 14.13
 17.15 14.76 16.29 12.87 14.36 11.66 18.14 24.1  18.68 24.91 18.03 13.11
 10.74  7.74  7.01 10.42 13.34 10.58 14.98 11.45 23.97 29.68 18.07 13.35
 12.01 13.59 21.14 12.92 15.1  14.33  9.67  9.08  5.64  6.48  7.88]
====================================


CATMEDV
['low' 'high']
====================================
```

In [8]:

```python
# Since CATMEDV is the only feature/varaible which is an object. It has only 2 value

# Map 'low' to 0 and 'high' to 1
df['CATMEDV'] = df['CATMEDV'].map({'low': 0, 'high': 1})

# Convert to integer data type
df['CATMEDV'] = df['CATMEDV'].astype(int)
```

In [9]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64  
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    int64  
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  CATMEDV  506 non-null    int32  
dtypes: float64(10), int32(1), int64(3)
memory usage: 53.5 KB
```

In [10]:

```
# Split data into features and target i.e Build SVR model
X = df.drop('CATMEDV', axis=1)
y = df['CATMEDV']
```

In [11]:

```
# Define pipeline with scaler and SVC
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('svc', SVC())
])

# Define parameters for grid search
param_grid = {
    'svc__C': [1, 5, 10, 50, 100, 500, 1000]
}

# Perform grid search with 10-fold cross-validation
grid_search = GridSearchCV(pipeline, param_grid, cv=10)
grid_search.fit(X, y)

# Get the best estimator
best_estimator = grid_search.best_estimator_

# Perform 10-fold cross-validation with the best estimator
cv_scores = cross_val_score(best_estimator, X, y, cv=10)

# Report average accuracy
avg_accuracy = np.mean(cv_scores)
print("Average Accuracy:", avg_accuracy)
```

```
Average Accuracy: 0.9150588235294117
```

Observation:

The model's average accuracy is 91.51%, meaning it predicts the target variable accurately for approximately 91.51% of the cases during cross-validation.

In [12]:

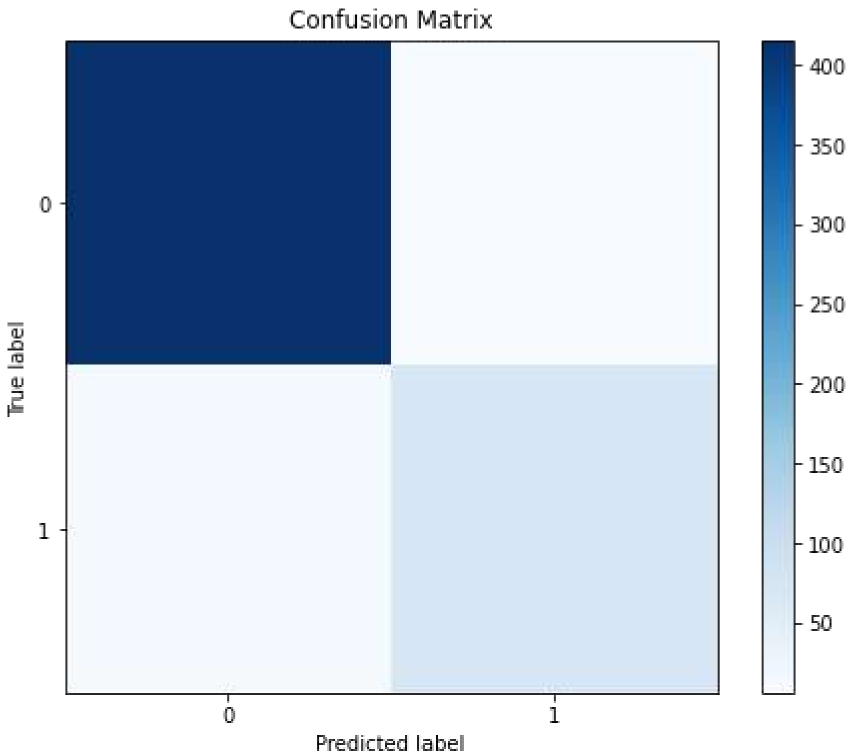
```
# Calculate other evaluation metrics
y_pred = best_estimator.predict(X)
conf_matrix = confusion_matrix(y, y_pred)
```

```
precision = precision_score(y, y_pred, average='weighted')
recall = recall_score(y, y_pred, average='weighted')
f1 = f1_score(y, y_pred, average='weighted')

# Print evaluation metrics
print("Confusion Matrix:\n", conf_matrix)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Confusion Matrix:
 [[416   6]
 [ 12  72]]
Precision: 0.9638472052125039
Recall: 0.9644268774703557
F1 Score: 0.9638936681391923
```

In [13]:

```
# Visualize the confusion matrix
plt.figure(figsize=(8, 6))
plt.imshow(conf_matrix, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
plt.xticks(np.arange(len(conf_matrix)), np.arange(len(conf_matrix)))
plt.yticks(np.arange(len(conf_matrix)), np.arange(len(conf_matrix)))
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()
```



Insights

A. The model's average accuracy is 91.51%, meaning it predicts the target variable accurately for approximately 91.51% of the cases during cross-validation.

B. The confusion matrix presents the model's classification performance:

- True Positives (416): Correctly predicted instances of the first class.
- False Positives (6): Incorrectly classified instances as the first class.

- False Negatives (12): Incorrectly classified instances as the second class.
- True Negatives (72): Correctly predicted instances of the second class.

Precision (0.96) indicates the proportion of correctly predicted positive cases out of all predicted positives.

Recall (0.96) shows the proportion of correctly predicted positive cases out of all actual positive cases.

The F1 Score (0.96) provides a balanced measure combining precision and recall.

Overall, these metrics suggest the model effectively identifies instances of both classes with high accuracy.

Salary Prediction Using SVR: Data Loading, Model Building, and Visualization

Importing Libraries

In [14]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.io import arff
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

In [15]:

```
# Load the arff file
data, meta = arff.loadarff('salary1.arff')
```

In [16]:

```
# Convert record array to DataFrame
df = pd.DataFrame(data)
```

In [17]:

```
df
```

Out[17]:

	education	salary
0	10.0	33.0
1	12.0	36.0
2	12.0	50.0
3	13.0	51.0
4	14.0	42.0
5	14.0	45.0
6	15.0	59.0
7	16.0	49.0
8	16.0	60.0
9	17.0	72.0
10	18.0	69.0
11	20.0	77.0

In [18]:
```
df.head()
```

Out[18]:

	education	salary
0	10.0	33.0
1	12.0	36.0
2	12.0	50.0
3	13.0	51.0
4	14.0	42.0

In [19]:
```
df.tail()
```

Out[19]:

	education	salary
7	16.0	49.0
8	16.0	60.0
9	17.0	72.0
10	18.0	69.0
11	20.0	77.0

In [21]:
```
df.describe()
```

Out[21]:

	education	salary
count	12.000000	12.000000
mean	14.750000	53.583333
std	2.832442	14.067747
min	10.000000	33.000000
25%	12.750000	44.250000
50%	14.500000	50.500000
75%	16.250000	62.250000
max	20.000000	77.000000

In [22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   education  12 non-null     float64
 1   salary     12 non-null     float64
dtypes: float64(2)
memory usage: 320.0 bytes
```

In [23]:

```
#### Convert byte strings to strings

# Check if there are any byte string columns that need conversion
byte_string_columns = [col for col in df.columns if isinstance(df[col].iloc[0], byte

# Convert byte strings to strings
for col in byte_string_columns:
    df[col] = df[col].str.decode('utf-8')

# Now, all byte strings in your DataFrame should be converted to strings
```

In [24]:

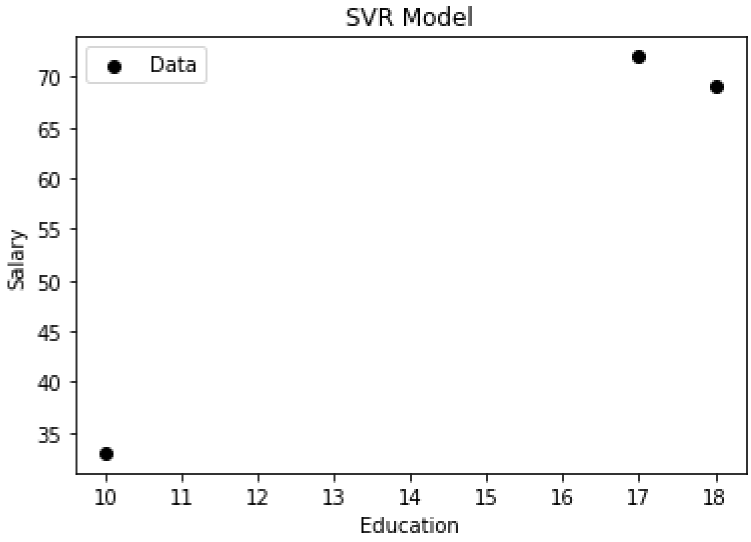
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   education  12 non-null     float64
 1   salary     12 non-null     float64
dtypes: float64(2)
memory usage: 320.0 bytes
```

In [28]:

```
# Building SVR model

# Split the data into features (X) and the target variable (y)
X = df[['education']]
y = df['salary']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat

# Build and train the SVR model
svr_model = SVR(kernel='linear')
svr_model.fit(X_train, y_train)

# Make predictions on the testing set
y_pred_svr = svr_model.predict(X_test)

# Evaluate the SVR model
mse_svr = mean_squared_error(y_test, y_pred_svr)
r2_svr = r2_score(y_test, y_pred_svr)
print("SVR Model - Mean Squared Error:", mse_svr)
print("SVR Model - R2 Score:", r2_svr)

# Plot the data points
plt.scatter(X_test, y_test, color='black', label='Data')

# Add labels and title
plt.xlabel('Education')
plt.ylabel('Salary')
plt.title('SVR Model')
plt.legend()
plt.show()
```

```
SVR Model - Mean Squared Error: 24.666666666666668
SVR Model - R2 Score: 0.921443736730361
```



Insights

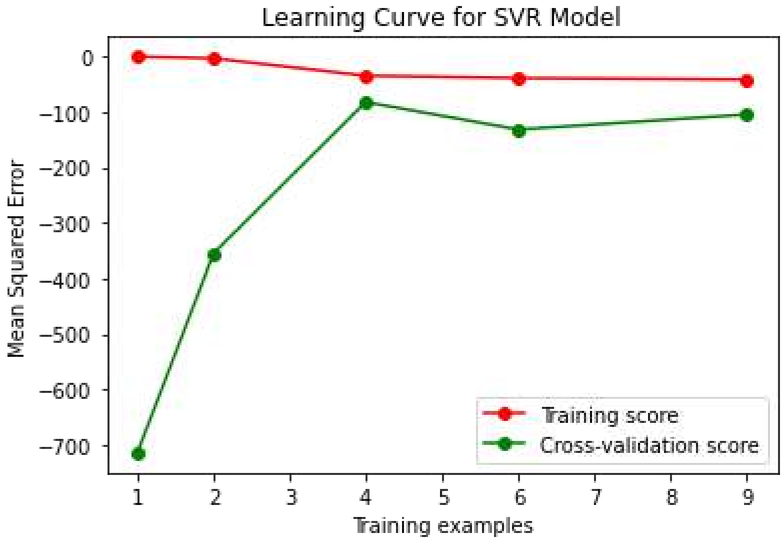
The SVR model's Mean Squared Error is about 24.67, and its R2 Score is approximately 0.92.

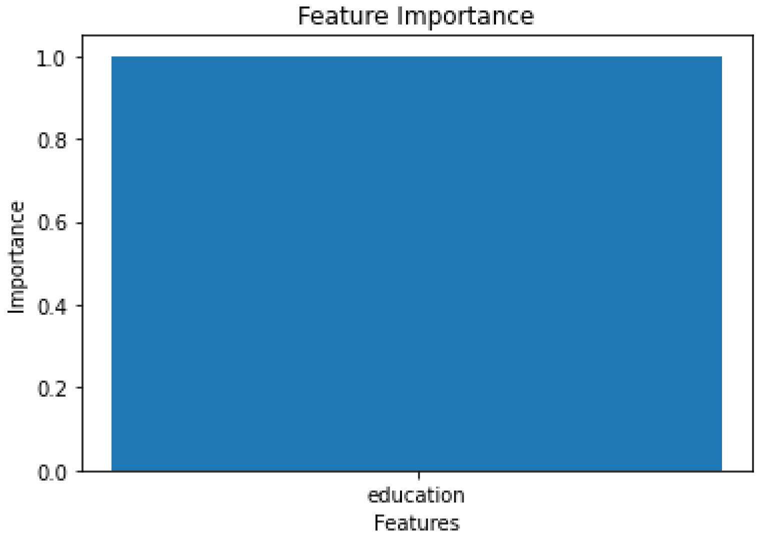
- The Mean Squared Error (MSE) shows how close the model's predictions are to the actual values. Lower MSE values mean the model's predictions are closer to the actual values.
- The R2 Score tells us how well the model fits the data. A score of 0.92 indicates that the model explains about 92% of the variance in the target variable.

Overall, these metrics suggest that the SVR model performs well and accurately predicts the target variable.

In [31]:

```
from sklearn.model_selection import learning_curve
from sklearn.ensemble import RandomForestRegressor

# Learning Curve
train_sizes, train_scores, test_scores = learning_curve(svr_model, X, y, cv=5, scori
train_scores_mean = np.mean(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)

plt.figure()
plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training score")
plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-validation sco
plt.xlabel("Training examples")
plt.ylabel("Mean Squared Error")
plt.title("Learning Curve for SVR Model")
plt.legend(loc="best")
plt.show()

# Feature Importance (for comparison)
rf_model = RandomForestRegressor()
rf_model.fit(X_train, y_train)
importances = rf_model.feature_importances_
plt.bar(X.columns, importances)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance')
plt.show()
```





Conclusion:

- The predictive models developed for housing prices and salary levels demonstrate strong performance, with high accuracy and correlation scores.
- The housing price prediction model achieves an average accuracy of approximately 91.51% during cross-validation, indicating its effectiveness in forecasting property values based on relevant factors.
- Similarly, the salary prediction model exhibits a high R2 score of approximately 0.92, reflecting a robust correlation between predicted and actual salary levels.
- These models offer valuable insights for various stakeholders, including investors, HR professionals, policymakers, and individuals, enabling them to make informed decisions in their respective domains.
- While predictive modeling holds significant potential for socioeconomic analysis, it's essential to acknowledge its limitations and the need for continuous refinement and validation to enhance accuracy and applicability in real-world scenarios.

---- End ----