Data Visualization and SVM BY selecting random features

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np
from sklearn.metrics import accuracy_score

# Specify the file path
file_path = '/content/drive/MyDrive/train.csv'

# Load the dataset
data = pd.read_csv(file_path)
print(data.head)

# Specify the columns to extract (replace with actual column names)
columns_to_extract = ['LotArea', 'MSZoning', 'SalePrice']  # Replace with your desired
column names

# Select the specific columns and first 20 rows
subset_data = data[columns_to_extract].head(100)
```

```
# Display the data as a table
print("Subset of the data (3 columns, 100 rows):")
display(subset_data)

# Visualize the data with plots
# 1. Bar Plot for one of the columns
plt.figure(figsize=(10, 6))
sns.barplot(data=subset_data, x='LotArea', y='SalePrice')  # Replace with actual column names
plt.title('Bar Plot of AreaLot vs SalePrice')
plt.xlabel('LotArea')
plt.ylabel('SalePrice')
plt.xticks(rotation=45)
plt.show()

# 2. Scatter Plot for two columns
plt.figure(figsize=(10, 6))
sns.scatterplot(data=subset_data, x='LotArea', y='SalePrice', hue='MSZoning', palette='viridis')  # Replace as needed
plt.title('Scatter Plot of Column1 vs Column3')
plt.xlabel('LotArea')
plt.ylabel('SalePrice')
plt.show()

# Select only numeric columns for the heatmap
numeric_data = subset_data.select_dtypes(include=['float64', 'int64'])

# Check if there are numeric columns
if not numeric_data.empty:
   # Create the heatmap
```

```python
    plt.figure(figsize=(8, 5))
    sns.heatmap(numeric_data.corr(), annot=True, cmap='coolwarm', fmt='.2f')
    plt.title('Correlation Heatmap of Selected Columns')
    plt.show()
else:
    print("No numeric columns available for heatmap.")


# Select numeric columns for features and one column for the target
columns_to_extract = ['GrLivArea', 'TotalBsmtSF', 'SalePrice']  # Update with other column names
subset_data = data[columns_to_extract].dropna().head(100)  # Ensure no missing values and use first 100 rows

# Prepare features (GrLivArea, TotalBsmtSF) and target (SalePrice)
features = subset_data[['GrLivArea', 'TotalBsmtSF']]  # Two numeric columns
target = subset_data['SalePrice']

# Bin SalePrice into categories: 'Low', 'High'
subset_data['PriceCategory'] = pd.cut(target, bins=2, labels=['Low', 'High'])
target_encoded = LabelEncoder().fit_transform(subset_data['PriceCategory'])

# Standardize numeric features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(features_scaled, target_encoded, test_size=0.3, random_state=42)

# Create and train Linear SVM model
```

```
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)

# Make predictions
y_pred = svm_model.predict(X_test)

# Evaluate the model
print("Classification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Visualize the linear hyperplane
plt.figure(figsize=(10, 6))

# Scatter plot of features with their labels
for i, label in enumerate(['Low', 'High']):
    plt.scatter(features_scaled[target_encoded == i, 0], features_scaled[target_encoded == i,
1], label=label)

# Plot the decision boundary
coef = svm_model.coef_[0]
intercept = svm_model.intercept_[0]
slope = -coef[0] / coef[1]
xx = np.linspace(features_scaled[:, 0].min(), features_scaled[:, 0].max())
yy = slope * xx - intercept / coef[1]
plt.plot(xx, yy, 'k--', label='Hyperplane')
coef = svm_model.coef_[0]
intercept = svm_model.intercept_[0]
```

```
# Define slope and intercept of the hyperplane
slope = -coef[0] / coef[1]
threshold_intercept = -intercept / coef[1]

print(f"Slope of Hyperplane: {slope}")
print(f"Threshold Intercept on Hyperplane: {threshold_intercept}")
plt.title("Linear SVM Decision Boundary")
plt.xlabel("GrLivArea (scaled)")
plt.ylabel("TotalBsmtSF (scaled)")
plt.legend()
plt.grid()
plt.show()

x1= subset_data['GrLivArea']
y = subset_data['SalePrice']

# Calculate correlation coefficient
r = np.corrcoef(x1, y)[0, 1]
print(f"Correlation Coefficient (GrLivArea vs SalePrice): {r:.2f}")

x2 = subset_data['TotalBsmtSF']
y = subset_data['SalePrice']
r = np.corrcoef(x2, y)[0, 1]
print(f"Correlation Coefficient (TotalBsmtSF vs SalePrice): {r:.2f}")

# Select only numeric columns
numeric_data = data.select_dtypes(include=['float64', 'int64'])
```

```
if 'SalePrice' in numeric_data.columns:
    # Calculate the correlation of each column with SalePrice
    saleprice_correlation = numeric_data.corr()['SalePrice'].sort_values(ascending=False)

    # Display the correlation values
    print("Correlation of each feature with SalePrice:")
    print(saleprice_correlation)
else:
    print("Column 'SalePrice' not found in the dataset.")

accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")
```

Output :

```
<bound method NDFrame.head of        Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0      1        60       RL         65.0     8450   Pave   NaN      Reg
1      2        20       RL         80.0     9600   Pave   NaN      Reg
2      3        60       RL         68.0    11250   Pave   NaN      IR1
3      4        70       RL         60.0     9550   Pave   NaN      IR1
4      5        60       RL         84.0    14260   Pave   NaN      IR1
...   ...       ...      ...          ...      ...    ...   ...      ...
1455  1456        60       RL         62.0     7917   Pave   NaN      Reg
1456  1457        20       RL         85.0    13175   Pave   NaN      Reg
1457  1458        70       RL         66.0     9042   Pave   NaN      Reg
1458  1459        20       RL         68.0     9717   Pave   NaN      Reg
1459  1460        20       RL         75.0     9937   Pave   NaN      Reg

     LandContour Utilities  ... PoolArea PoolQC  Fence MiscFeature MiscVal  \
0            Lvl    AllPub  ...        0    NaN    NaN         NaN       0
```

1 Lvl AllPub ... 0 NaN NaN NaN 0

2 Lvl AllPub ... 0 NaN NaN NaN 0

3 Lvl AllPub ... 0 NaN NaN NaN 0

4 Lvl AllPub ... 0 NaN NaN NaN 0

... ...

1455 Lvl AllPub ... 0 NaN NaN NaN 0

1456 Lvl AllPub ... 0 NaN MnPrv NaN 0

1457 Lvl AllPub ... 0 NaN GdPrv Shed 2500

1458 Lvl AllPub ... 0 NaN NaN NaN 0

1459 Lvl AllPub ... 0 NaN NaN NaN 0

MoSold YrSold SaleType SaleCondition SalePrice

0 2 2008 WD Normal 208500

1 5 2007 WD Normal 181500

2 9 2008 WD Normal 223500

3 2 2006 WD Abnorml 140000

4 12 2008 WD Normal 250000

... ...

1455 8 2007 WD Normal 175000

1456 2 2010 WD Normal 210000

1457 5 2010 WD Normal 266500

1458 4 2010 WD Normal 142125

1459 6 2008 WD Normal 147500

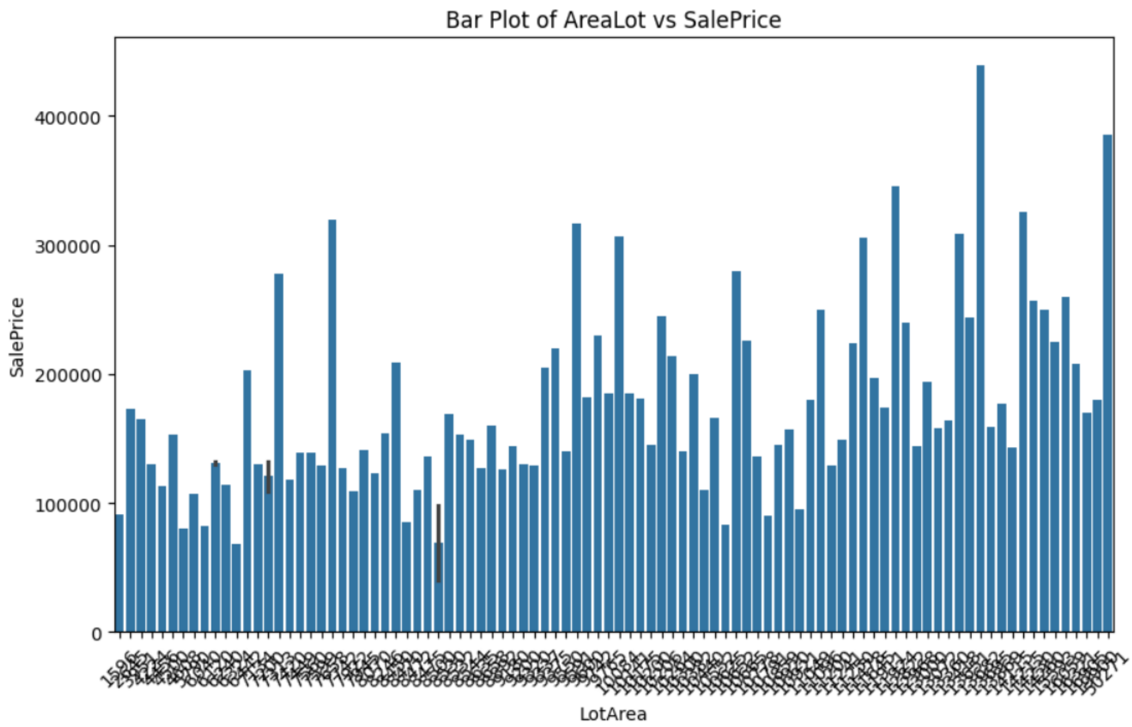
[1460 rows x 81 columns]>

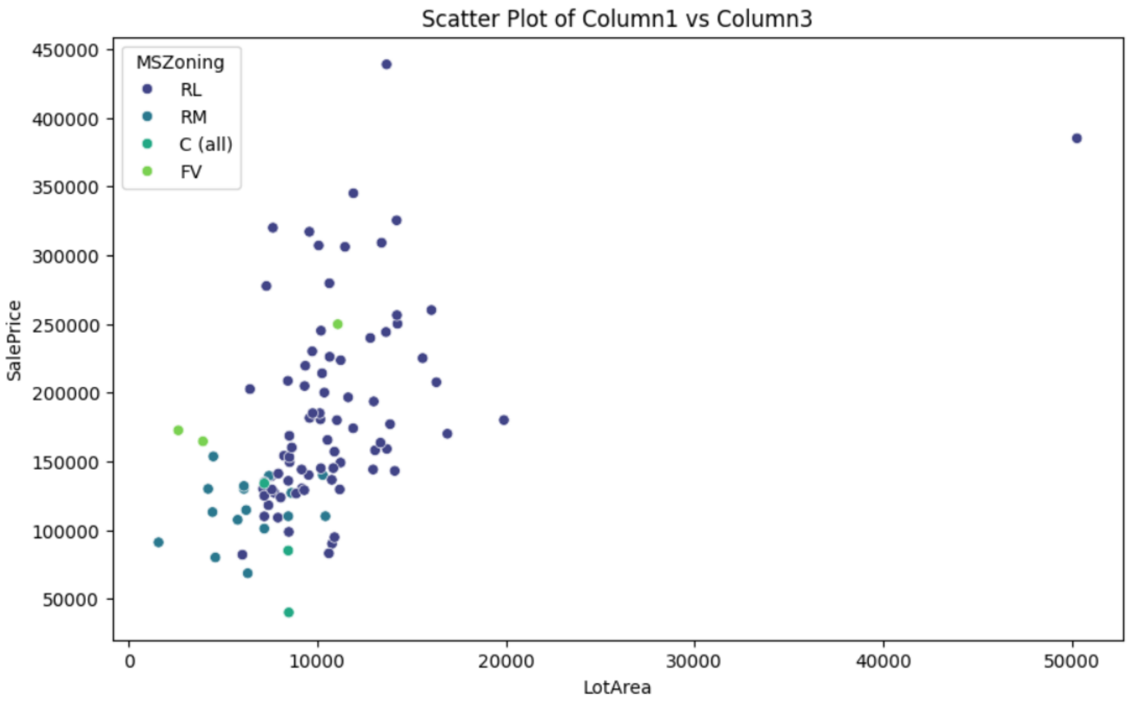
Subset of the data (3 columns, 100 rows):

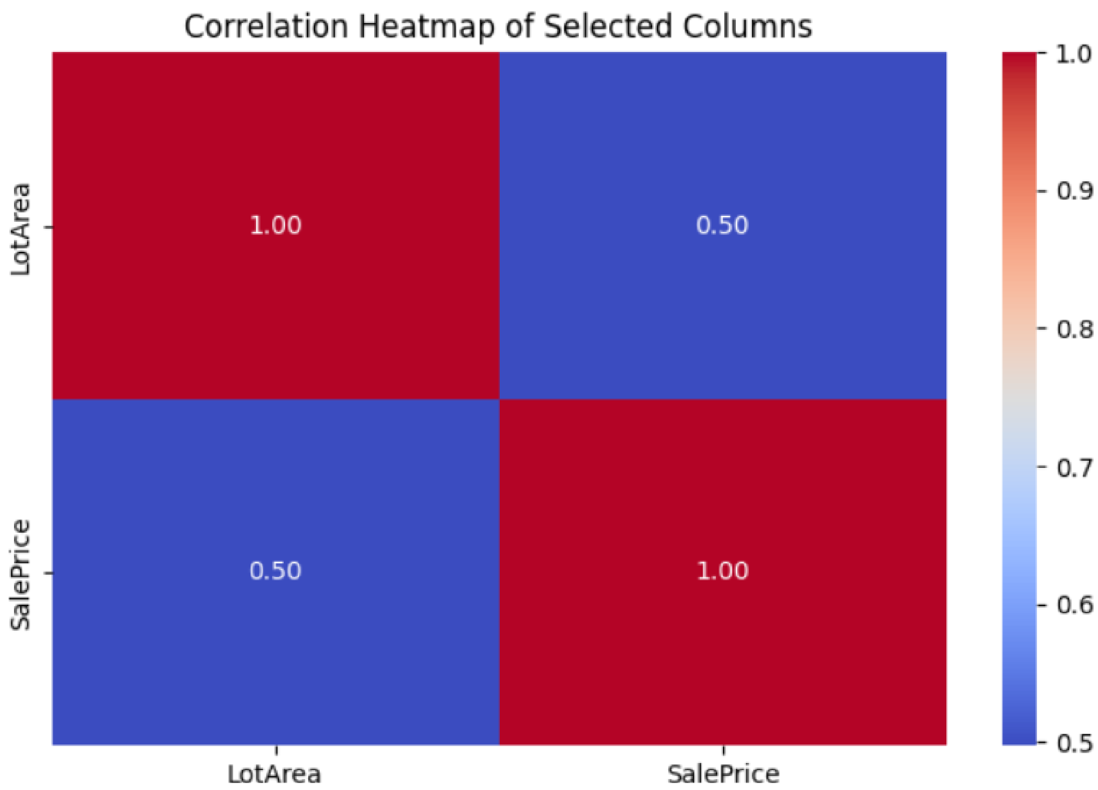
	LotArea	**MSZoning**	**SalePrice**
0	8450	RL	208500

	LotArea	MSZoning	SalePrice
1	9600	RL	181500
2	11250	RL	223500
3	9550	RL	140000
4	14260	RL	250000
...	...	...	...
95	9765	RL	185000
96	10264	RL	214000
97	10921	RL	94750
98	10625	RL	83000
99	9320	RL	128950

100 rows × 3 columns

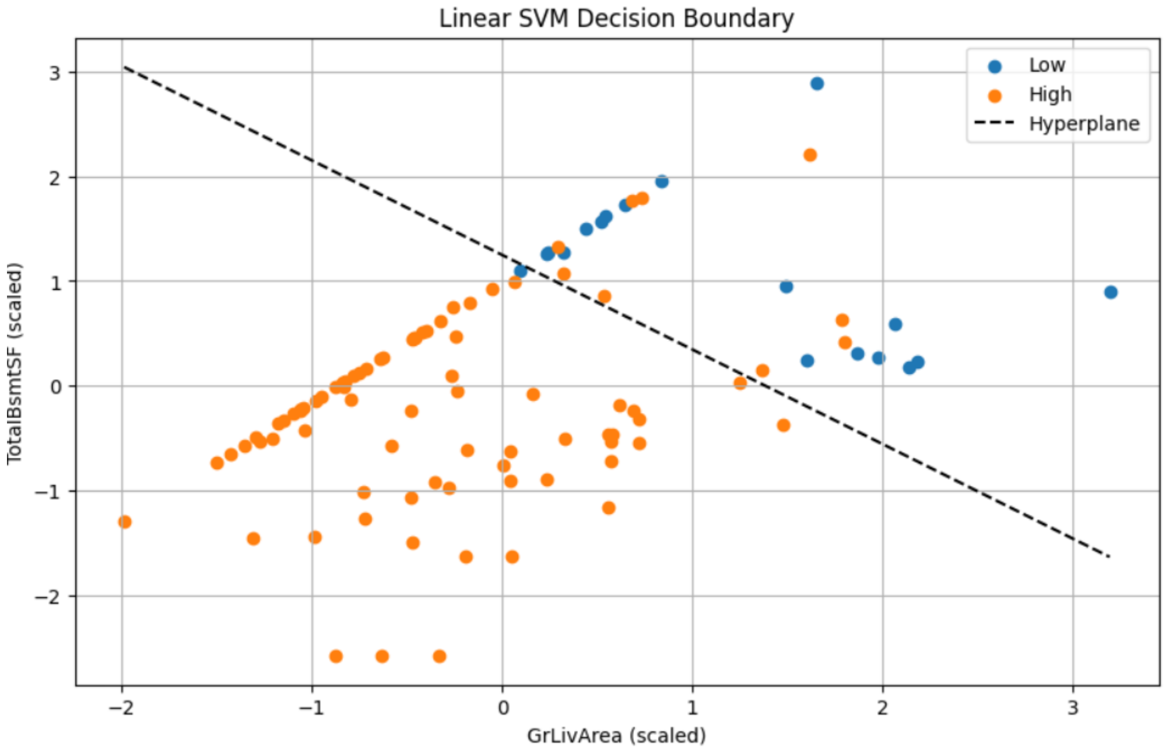






Classification Report:

	precision	recall	f1-score	support
0	0.50	1.00	0.67	4
1	1.00	0.85	0.92	26
accuracy			0.87	30
macro avg	0.75	0.92	0.79	30
weighted avg	0.93	0.87	0.88	30



Confusion Matrix:

```
[[ 4  0]
 [ 4 22]]
```

Slope of Hyperplane: -0.9036829820979518

Threshold Intercept on Hyperplane: 1.2492900768016308

Correlation Coefficient of each column with output to get better Accuracy

Correlation Coefficient (GrLivArea vs SalePrice): 0.74

Correlation Coefficient (TotalBsmtSF vs SalePrice): 0.62

Correlation of each feature with SalePrice:

SalePrice 1.000000

OverallQual 0.790982

GrLivArea 0.708624

GarageCars 0.640409

GarageArea 0.623431

TotalBsmtSF 0.613581

1stFlrSF 0.605852

FullBath 0.560664

TotRmsAbvGrd 0.533723

YearBuilt 0.522897

YearRemodAdd 0.507101

GarageYrBlt 0.486362

MasVnrArea 0.477493

Fireplaces 0.466929

BsmtFinSF1 0.386420

LotFrontage 0.351799

WoodDeckSF 0.324413

2ndFlrSF 0.319334

OpenPorchSF 0.315856

HalfBath 0.284108

LotArea 0.263843

BsmtFullBath 0.227122

BsmtUnfSF 0.214479

BedroomAbvGr   0.168213

ScreenPorch   0.111447

PoolArea   0.092404

MoSold   0.046432

3SsnPorch   0.044584

BsmtFinSF2   -0.011378

BsmtHalfBath   -0.016844

MiscVal   -0.021190

Id   -0.021917

LowQualFinSF   -0.025606

YrSold   -0.028923

OverallCond   -0.077856

MSSubClass   -0.084284

EnclosedPorch   -0.128578

KitchenAbvGr   -0.135907

Name: SalePrice, dtype: float64

Model Accuracy: 0.87

Highest Accuracy By choosing the most correlated colums as input

Select numeric columns for features and one column for the target with highesh accuracy

columns_to_extract = ['OverallQual','GrLivArea','SalePrice'] # Update with other column names

subset_data = data[columns_to_extract].dropna().head(100) # Ensure no missing values and use first 100 rows

Prepare features (GrLivArea, TotalBsmtSF) and target (SalePrice)

```
features = subset_data[['OverallQual','GrLivArea']]  # Two numeric columns
target = subset_data['SalePrice']

# Bin SalePrice into categories: 'Low', 'High'
subset_data['PriceCategory'] = pd.cut(target, bins=2, labels=['Low', 'High'])
target_encoded = LabelEncoder().fit_transform(subset_data['PriceCategory'])

# Standardize numeric features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(features_scaled, target_encoded,
test_size=0.3, random_state=42)

# Create and train Linear SVM model
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)

# Make predictions
y_pred = svm_model.predict(X_test)

# Evaluate the model
print("Classification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Visualize the linear hyperplane
```

```
plt.figure(figsize=(10, 6))

# Scatter plot of features with their labels
for i, label in enumerate(['Low', 'High']):
    plt.scatter(features_scaled[target_encoded == i, 0], features_scaled[target_encoded == i,
1], label=label)

# Plot the decision boundary
coef = svm_model.coef_[0]
intercept = svm_model.intercept_[0]
slope = -coef[0] / coef[1]
xx = np.linspace(features_scaled[:, 0].min(), features_scaled[:, 0].max())
yy = slope * xx - intercept / coef[1]
plt.plot(xx, yy, 'k--', label='Hyperplane')
coef = svm_model.coef_[0]
intercept = svm_model.intercept_[0]

# Define slope and intercept of the hyperplane
slope = -coef[0] / coef[1]
threshold_intercept = -intercept / coef[1]

print(f"Slope of Hyperplane: {slope}")
print(f"Threshold Intercept on Hyperplane: {threshold_intercept}")
plt.title("Linear SVM Decision Boundary")
plt.xlabel("GrLivArea (scaled)")
plt.ylabel("OverallQual (scaled)")
plt.legend()
plt.grid()
plt.show()
accuracy = accuracy_score(y_test, y_pred)
```

```
print(f"Model Accuracy: {accuracy:.2f}")
```

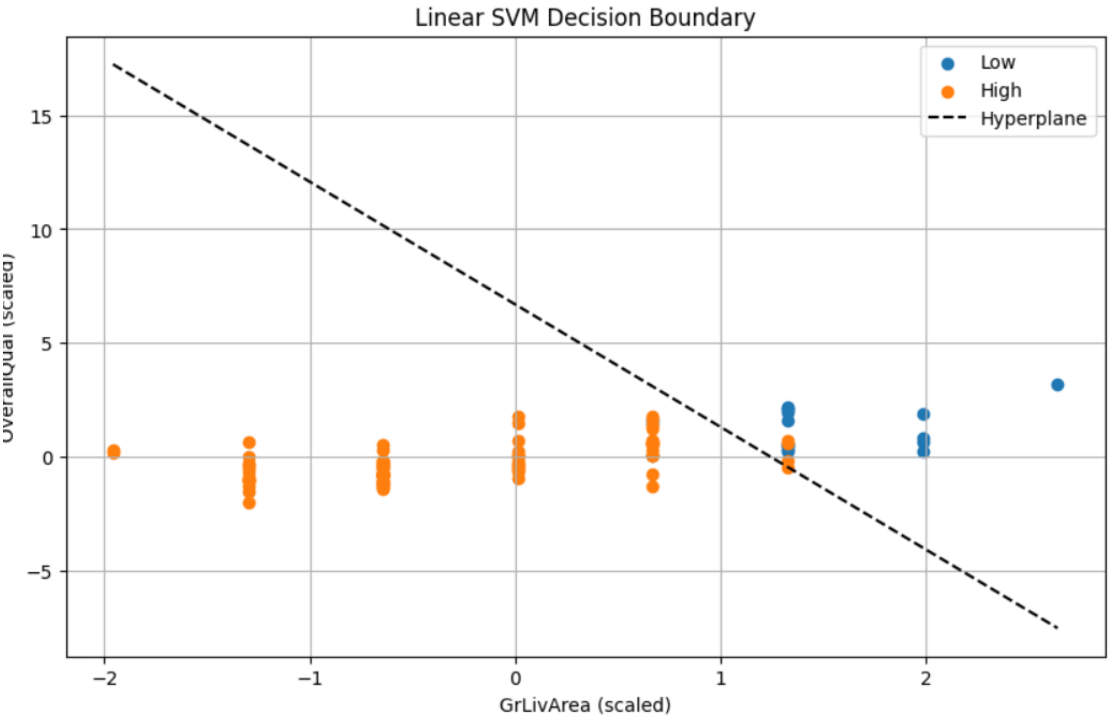
Output :

```
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.75      0.75         4
           1       0.96      0.96      0.96        26

    accuracy                           0.93        30
   macro avg       0.86      0.86      0.86        30
weighted avg       0.93      0.93      0.93        30


Confusion Matrix:
[[ 3  1]
 [ 1 25]]
Slope of Hyperplane: -5.383770452884379
Threshold Intercept on Hyperplane: 6.6949558734721775
```



Model Accuracy: 0.93

Output when taking less correlated inputs

Select numeric columns for features and one column for the target

columns_to_extract = ['LotFrontage','BsmtFinSF1','SalePrice'] # Update with other column names

subset_data = data[columns_to_extract].dropna().head(100) # Ensure no missing values and use first 100 rows

Prepare features (GrLivArea, TotalBsmtSF) and target (SalePrice)

features = subset_data[['LotFrontage','BsmtFinSF1']] # Two numeric columns

target = subset_data['SalePrice']

Bin SalePrice into categories: 'Low', 'High'

subset_data['PriceCategory'] = pd.cut(target, bins=2, labels=['Low', 'High'])

target_encoded = LabelEncoder().fit_transform(subset_data['PriceCategory'])

```
# Standardize numeric features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(features_scaled, target_encoded,
test_size=0.3, random_state=42)

# Create and train Linear SVM model
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)

# Make predictions
y_pred = svm_model.predict(X_test)

# Evaluate the model
print("Classification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Visualize the linear hyperplane
plt.figure(figsize=(10, 6))

# Scatter plot of features with their labels
for i, label in enumerate(['Low', 'High']):
    plt.scatter(features_scaled[target_encoded == i, 0], features_scaled[target_encoded == i,
1], label=label)
```

```
# Plot the decision boundary
coef = svm_model.coef_[0]
intercept = svm_model.intercept_[0]
slope = -coef[0] / coef[1]
xx = np.linspace(features_scaled[:, 0].min(), features_scaled[:, 0].max())
yy = slope * xx - intercept / coef[1]
plt.plot(xx, yy, 'k--', label='Hyperplane')
coef = svm_model.coef_[0]
intercept = svm_model.intercept_[0]

# Define slope and intercept of the hyperplane
slope = -coef[0] / coef[1]
threshold_intercept = -intercept / coef[1]

print(f"Slope of Hyperplane: {slope}")
print(f"Threshold Intercept on Hyperplane: {threshold_intercept}")
plt.title("Linear SVM Decision Boundary")
plt.xlabel("LotFrontage (scaled)")
plt.ylabel("BsmtFinSF1 (scaled)")
plt.legend()
plt.grid()
plt.show()
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")
```

Output :

```
Classification Report:
           precision    recall  f1-score   support

        0       0.00      0.00      0.00         9
```

```
           1       0.70      1.00      0.82        21

    accuracy                           0.70        30
   macro avg       0.35      0.50      0.41        30
weighted avg       0.49      0.70      0.58        30
```

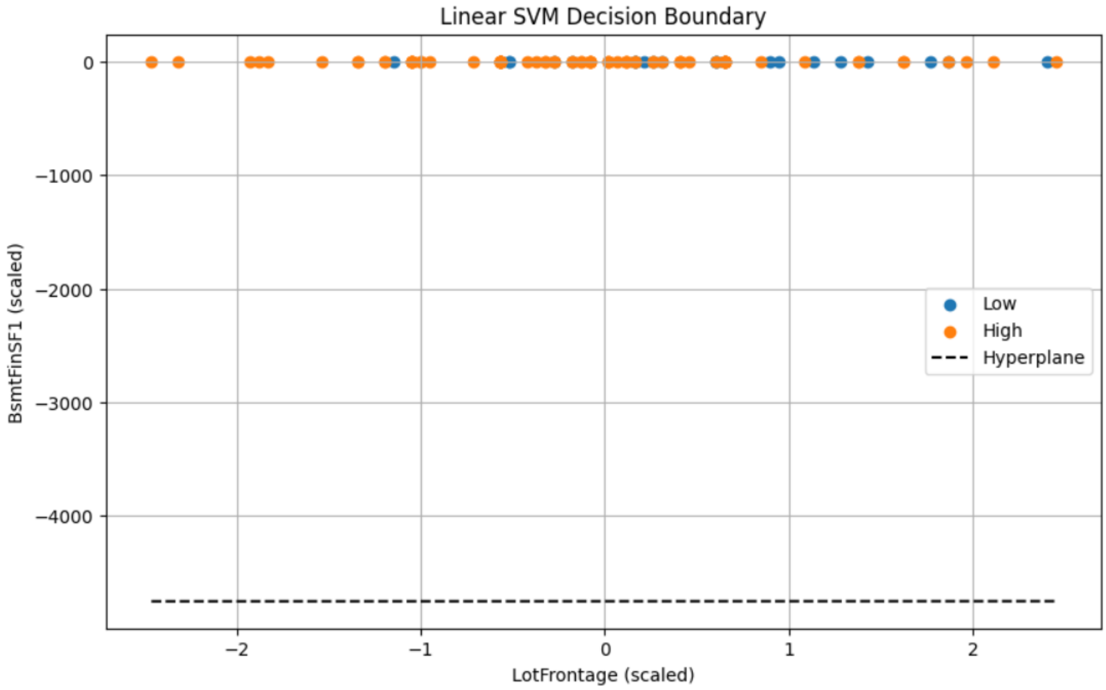
Confusion Matrix:

```
[[ 0  9]
 [ 0 21]]
```

Slope of Hyperplane: 0.2311629509675174

Threshold Intercept on Hyperplane: -4753.151100667716

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```



Model Accuracy: 0.70