VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Hashu Advani Memorial Complex, Collector's Colony, R C Marg, Chembur, Mumbai-400074

V.E.S.
Since 1962

Department of Artificial Intelligence and Data Science

Subject: ML **Class:** D11AD **Semester:** VI

<table>
<tr><td>Roll No.:
26</td><td colspan="3">Name:
Dyotak Kachare</td></tr>
<tr><td>Exp No.:
2</td><td colspan="3">Title:
Implement linear regression</td></tr>
<tr><td>DOP:</td><td>24/01</td><td>DOS:</td><td>31/01</td></tr>
<tr><td>GRADE</td><td></td><td>LAB OUTCOME:
LO2</td><td>SIGNATURE:</td></tr>
</table>

Name: Class: Div: Roll No:

Subject: Topic: Date: Page No:

ML Experiment - 2

Aim -

Implementing Linear Regression for a given data set.

Theory -

Linear regression is a type of linear model that is considered most basic and commonly used predictive algo.

A linear model assumes a linear rel between input variables (x) and output variable (y).

Regularization -

Regularization is implemented to avoid overfitting of the data, especially when there is a large variance between train and test performance.

Lasso Regression -

Lasso is short for Least Absolute Shrinkage and Selection Operator and model selection.

This variation differs from ridge regression only penalizing the high coefficient.

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right) + \lambda \sum_{j=1}^{n} |\beta_j|$$

Ridge regression

RSS is modified by adding the shrinkage quantity. λ is the tuning parameter that decides how much we want to penalize the flexibility of our model.

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right) + \lambda \sum_{j=1}^{p} \beta_j^2$$

Conclusion

Thus we have implemented linear regression and applied regularization technique to improve performance of model.

[1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error
```

[2]:
```python
df = pd.read_csv('./mi_csk.csv')
df.head()
```

[2]:

	match_id	mi_batting	inning	over	ball	total_runs	team_score
0	1	0	1	1	1	1	1
1	1	0	1	1	2	0	1
2	1	0	1	1	3	0	1
3	1	0	1	1	4	0	1
4	1	0	1	1	5	4	5

[3]:
```python
df['ball'] = df['ball'] + ((df['over']-1)*6)
```

[4]:
```python
df = df[(df['match_id'] == 9) | (df['match_id'] == 11)]
```

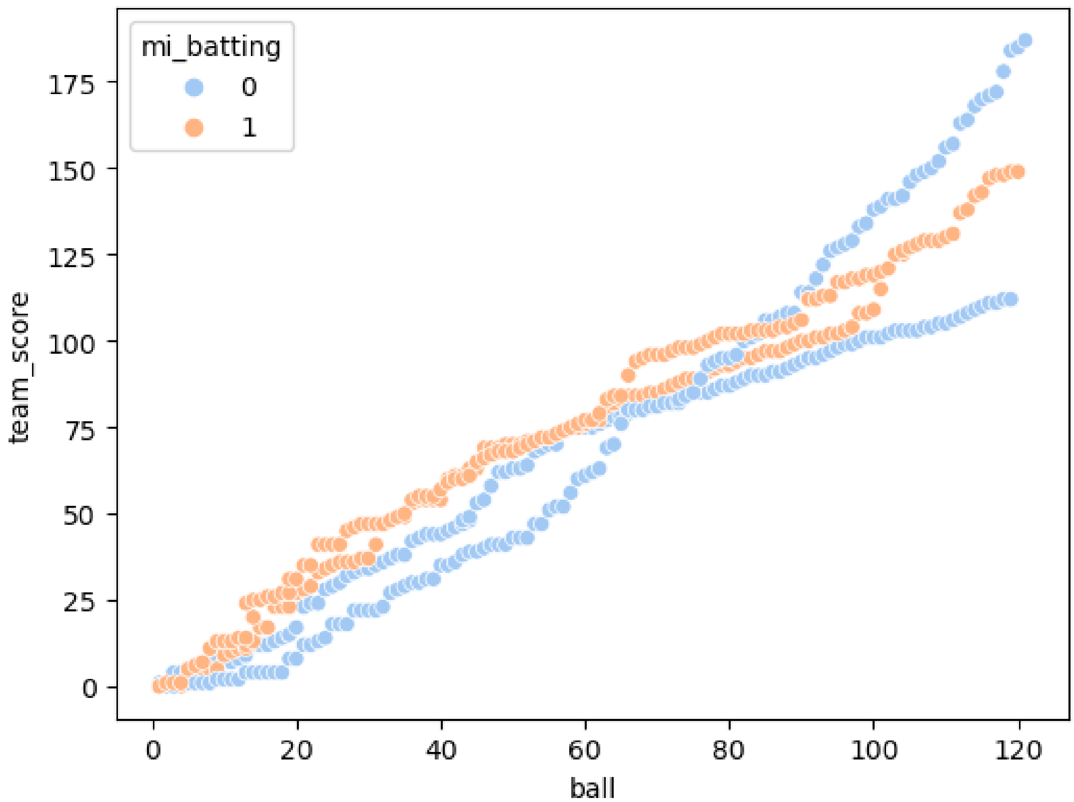
[5]:
```python
df = df.drop(['match_id', 'over'], axis=1)
df
df.head()
```

[5]:

	mi_batting	inning	ball	total_runs	team_score
1975	0	1	1	1	1
1976	0	1	2	0	1
1977	0	1	3	3	4
1978	0	1	4	0	4
1979	0	1	5	0	4

[6]:
```python
sns.scatterplot(data=df, x="ball", y="team_score", hue="mi_batting",
    ↪palette="pastel")
```

[6]: <Axes: xlabel='ball', ylabel='team_score'>



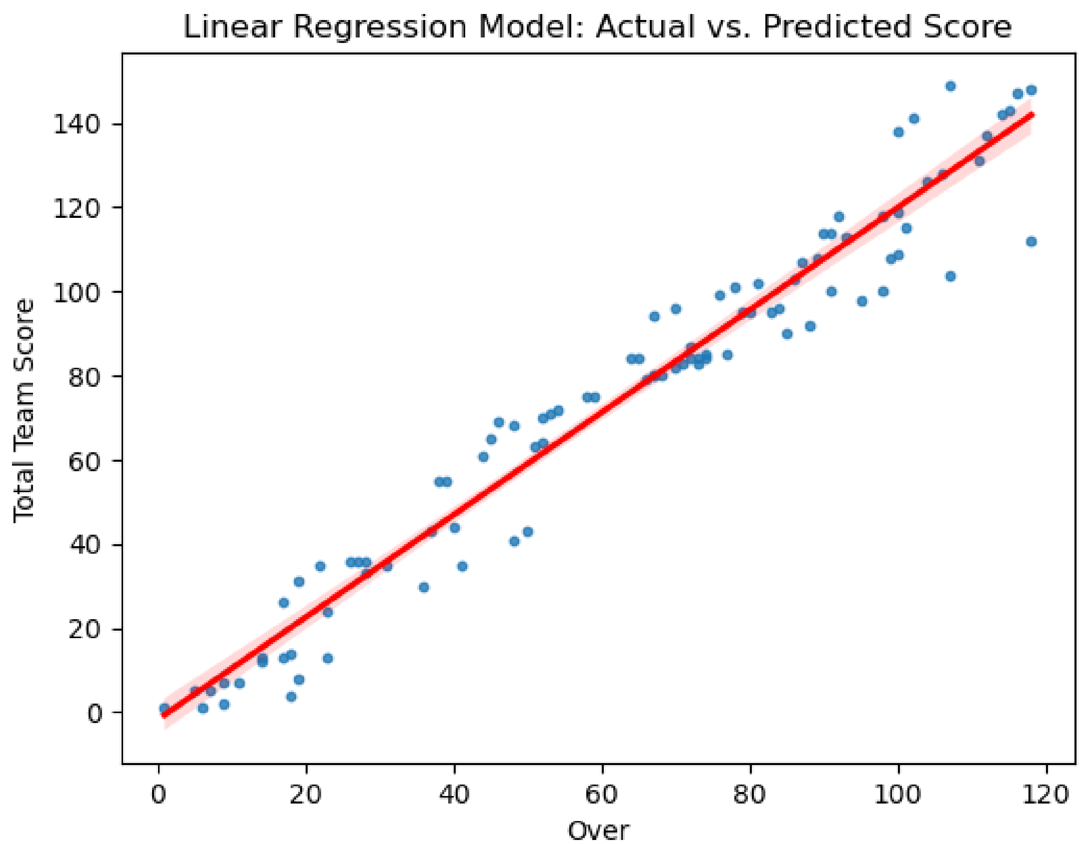
[7]:
```
features = df.drop("team_score", axis=1)
target = df["team_score"]

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.
    ↪2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
```

[7]: LinearRegression()

[8]:
```
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

Mean Squared Error: 71.17024747577479

[9]:
```
sns.regplot(x=X_test['ball'], y=y_test, scatter_kws={'s': 10}, line_kws={'color':
    ↪'red'})
plt.xlabel('Over')
plt.ylabel('Total Team Score')
plt.title('Linear Regression Model: Actual vs. Predicted Score')
plt.show()
```



Applying cross-validation

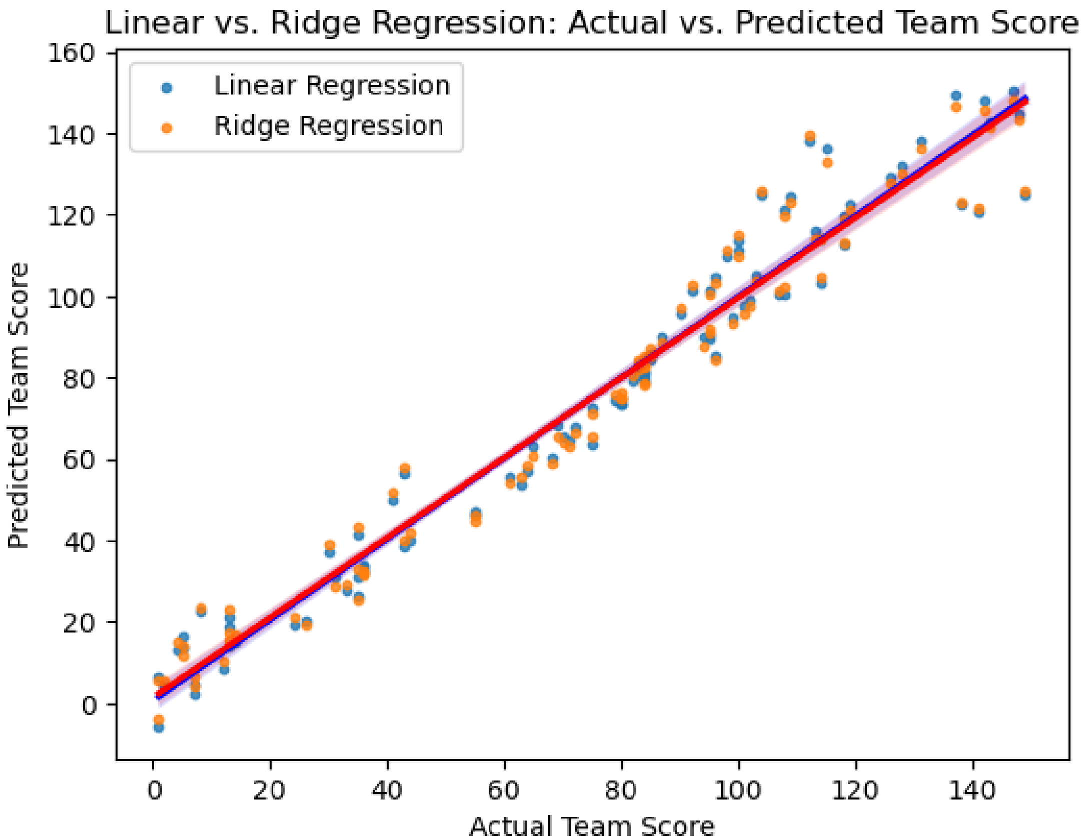
0.0.1 Ridge Regression

[10]:
```
alpha = 100
ridge_model = Ridge(alpha=alpha)
ridge_model.fit(X_train, y_train)

ridge_y_pred = ridge_model.predict(X_test)

mse = mean_squared_error(y_test, ridge_y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 69.12664235887162
```

[11]:
```
sns.regplot(x=y_test, y=y_pred, scatter_kws={'s': 10}, line_kws={'color': 'blue'},
 ↪label='Linear Regression')
sns.regplot(x=y_test, y=ridge_y_pred, scatter_kws={'s': 10}, line_kws={'color':
 ↪'red'}, label='Ridge Regression')

plt.xlabel('Actual Team Score')
plt.ylabel('Predicted Team Score')
plt.title('Linear vs. Ridge Regression: Actual vs. Predicted Team Score')
```

```
plt.legend()
plt.show()
```



0.0.2 Lasso Regression

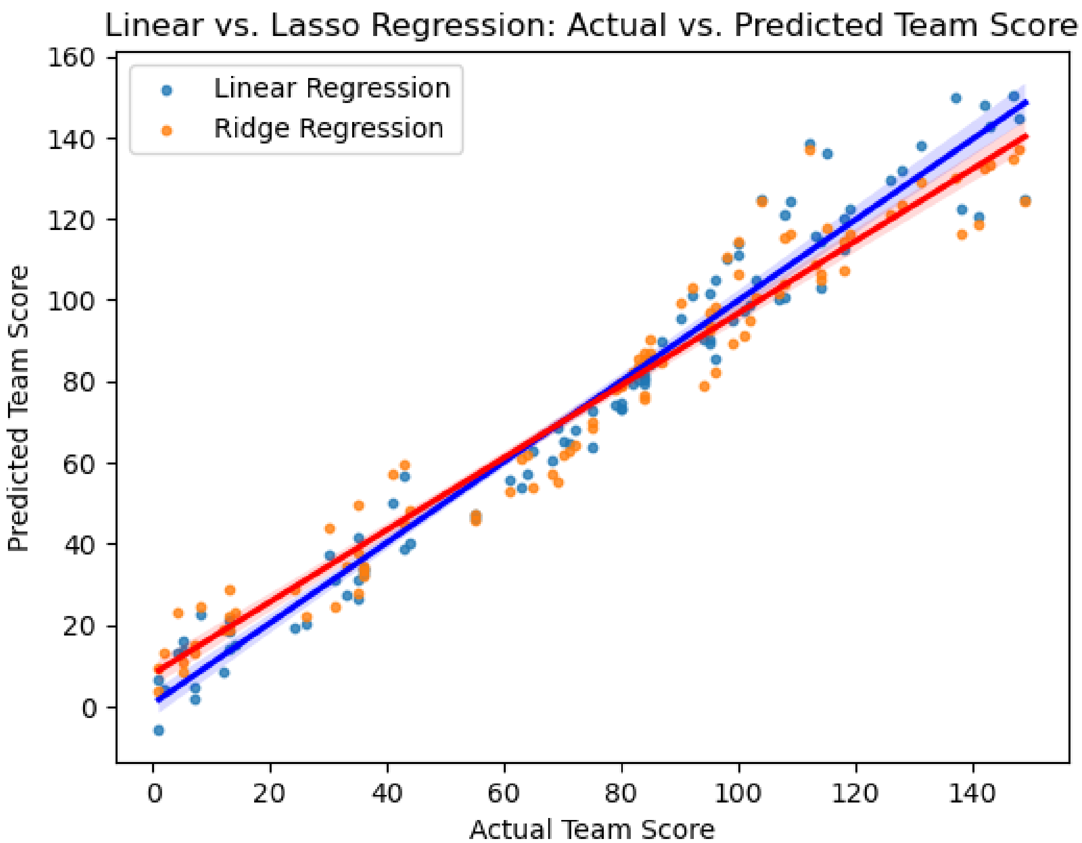
[12]:
```
alpha = 100
lasso_model = Lasso(alpha=alpha)
lasso_model.fit(X_train, y_train)

lasso_y_pred = lasso_model.predict(X_test)
```

[13]:
```
mse_lasso = mean_squared_error(y_test, lasso_y_pred)
print(f'Mean Squared Error (Lasso): {mse_lasso}')
```

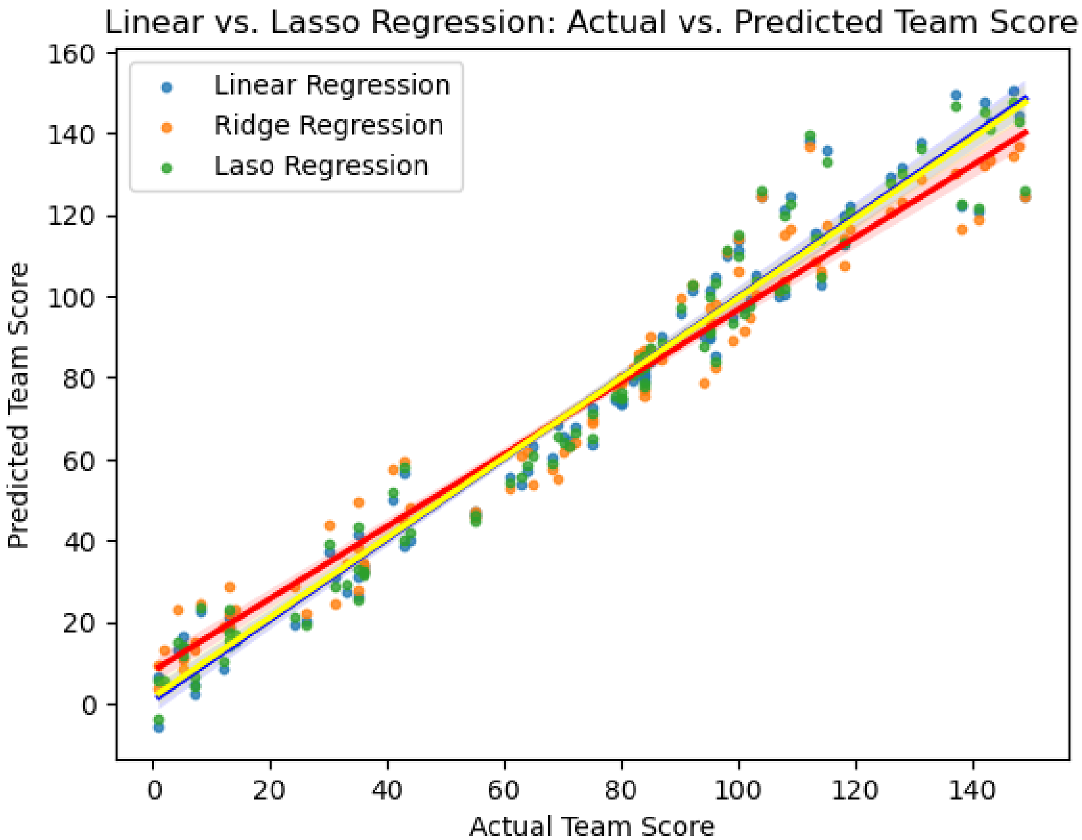
```
Mean Squared Error (Lasso): 89.71878185499247
```

[14]:
```
sns.regplot(x=y_test, y=y_pred, scatter_kws={'s': 10}, line_kws={'color': 'blue'},
    label='Linear Regression')
sns.regplot(x=y_test, y=lasso_y_pred, scatter_kws={'s': 10}, line_kws={'color':
    'red'}, label='Ridge Regression')

plt.xlabel('Actual Team Score')
plt.ylabel('Predicted Team Score')
```

```
plt.title('Linear vs. Lasso Regression: Actual vs. Predicted Team Score')
plt.legend()
plt.show()
```




```
[16]: sns.regplot(x=y_test, y=y_pred, scatter_kws={'s': 10}, line_kws={'color': 'blue'},
        label='Linear Regression')
      sns.regplot(x=y_test, y=lasso_y_pred, scatter_kws={'s': 10}, line_kws={'color':
        'red'}, label='Ridge Regression')
      sns.regplot(x=y_test, y=ridge_y_pred, scatter_kws={'s': 10}, line_kws={'color':
        'yellow'}, label='Laso Regression')

      plt.xlabel('Actual Team Score')
      plt.ylabel('Predicted Team Score')
      plt.title('Linear vs. Lasso Regression: Actual vs. Predicted Team Score')
      plt.legend()
      plt.show()
```


Linear vs. Lasso Regression: Actual vs. Predicted Team Score
Linear Regression
Ridge Regression
Laso Regression
Predicted Team Score
Actual Team Score
160
140
120
100
80
60
40
20
0
0
20
40
60
80
100
120
140