Name : Khan Kashif Aref

Roll no : 25

Class : SY-C-CSD

Course : data science

TOPIC :

REGRESSION MODEL WITH DATA SET

INTRODUCTION

General Steps for Regression Modeling and Visualization:

Define the Problem:

Clearly state the variable you want to predict (dependent variable) and the factors that might influence it (independent variables).
In this case, we're interested in predicting bowling average (dependent variable) based on factors like innings played, balls bowled, runs conceded, wickets taken, etc. (independent variables).

Data Acquisition and Preprocessing (Imagine you have data):

Imagine you have a table containing cricket bowler data.
You would use tools like pandas (in Python) to import this data into a structured format.
This might involve handling missing values, outliers, and ensuring data quality.

Feature Engineering (Imagine you have data):

This involves creating new features from existing ones that might better capture the relationship between variables.
For bowling data, you could create features like strike rate (wickets per over) or economy rate (runs conceded per over).

Model Selection and Training (Imagine you have data):

Choose a regression model that suits your data and problem. Common choices include linear regression, random forest regression, etc.
Split the data into training and testing sets. The training set is used to train the model, and the testing set is used to evaluate its performance on unseen data.
Train the model on the training data.

Visualization (Imagine you have data):

Create plots to visualize the relationship between the predicted and actual values of the dependent variable.
A common plot is the actual vs. predicted values, which helps identify how well the model captures the underlying trend.
You can also plot other relevant features against the dependent variable to understand the model's predictions.

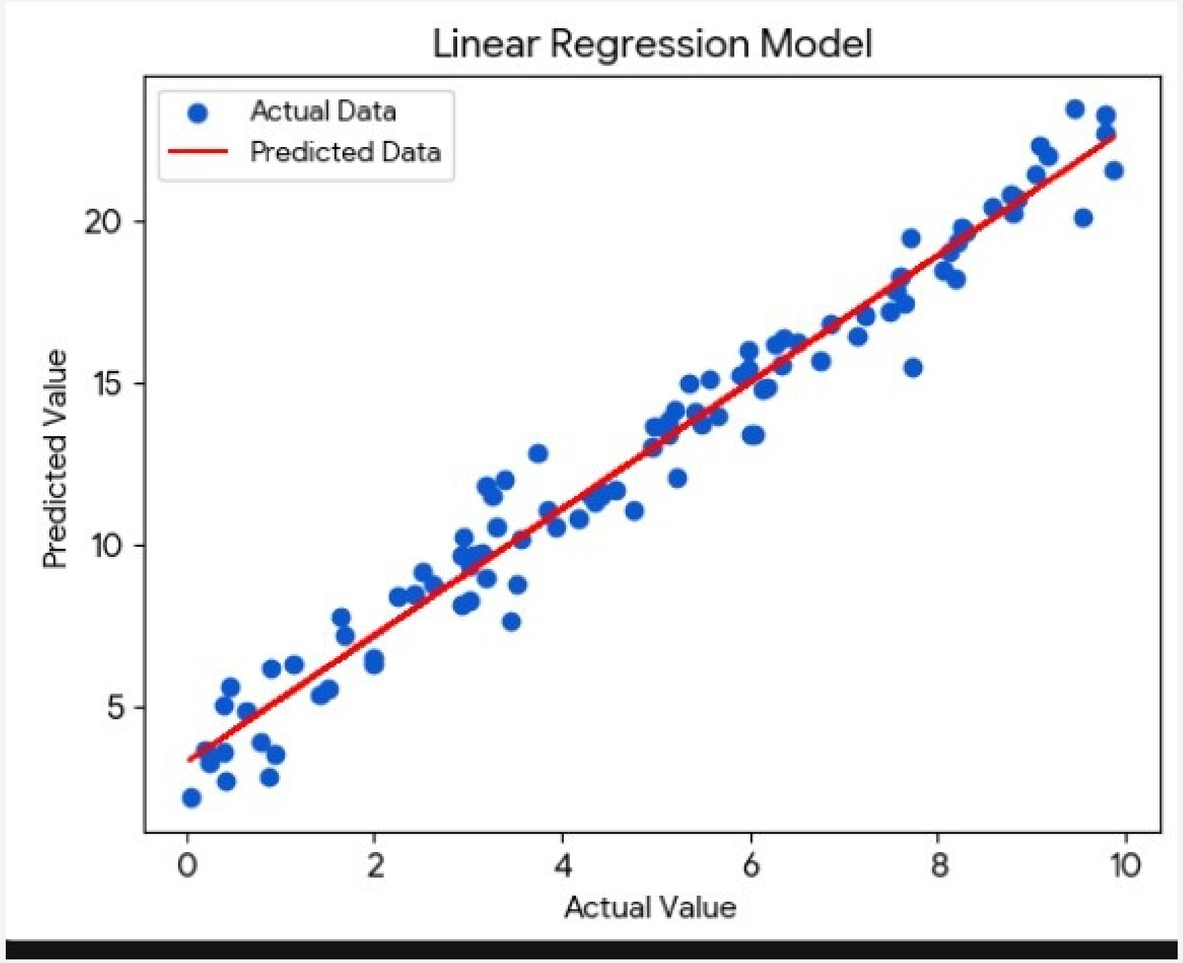
code snippet:

Imports matplotlib.pyplot as plt: This line imports the plotting functionalities from Matplotlib and assigns them the alias plt for convenience.
Plots the scatter graph: plt.scatter(y_test, y_pred) creates a scatter plot where each point represents a data sample from the testing set. The x-axis represents the actual bowling average (y_test), and the y-axis represents the predicted bowling average (y_pred).
Labels and title: The code adds labels to the x and y-axis using plt.xlabel and plt.ylabel. It also sets a title for the plot using plt.title.
Displays the plot: Finally, plt.show() displays the generated scatter plot.

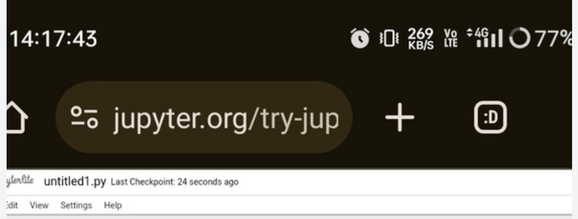
Graph : Actual vs Predicted Bowling Average

This is a scatter plot where each point represents a bowler from the testing set.
The x-axis represents the bowler's actual bowling average.
The y-axis represents the bowling average predicted by the linear regression model.
Ideally, the points would be clustered around a diagonal line from the bottom left to the top right corner. This indicates that the model's predictions are close to the actual values.
If the points are scattered randomly, it suggests the model isn't effectively capturing the relationship between the features and bowling average.
If there's a clear curve or pattern in the point distribution, it might indicate a non-linear relationship that a linear regression model might not handle well.
By analyzing this graph, you can get a sense of the model's fit and identify potential areas for improvement.

model


Linear Regression Model
Actual Data
Predicted Data
Predicted Value
Actual Value
0
2
4
6
8
10
5
10
15
20

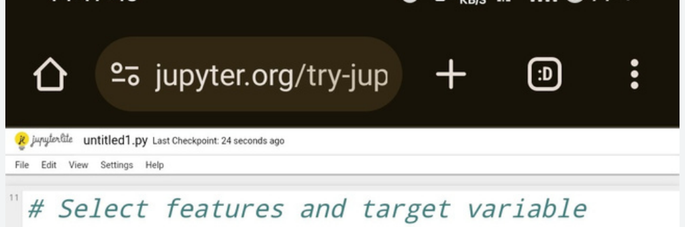
SK Warne (AUS)	1992-2007	145	273	40705	6784.1	1761	17995	**708**	8/71	25.41
JM Anderson (ENG)	2003-2024	187	348	39877	6646.1	1720	18569	**700**	7/42	26.52
A Kumble (IND)	1990-2008	132	236	40850	6808.2	1576	18355	**619**	10/74	29.65
SCJ Broad (ENG)	2007-2023	167	309	33698	5616.2	1304	16719	**604**	8/15	27.68
GD McGrath (AUS)	1993-2007	124	243	29248	4874.4	1470	12186	**563**	8/24	21.64
NM Lyon (AUS)	2011-2024	129	242	32761	5460.1	1044	16052	**530**	8/50	30.28
CA Walsh (WI)	1984-2001	132	242	30019	5003.1	1144	12688	**519**	7/37	24.44
R Ashwin (IND)	2011-2024	100	189	26166	4361.0	889	12255	**516**	7/59	23.75
DW Steyn (SA)	2004-2019	93	171	18608	3101.2	660	10077	**439**	7/51	22.95
N Kapil Dev (IND)	1978-1994	131	227	27740	4623.2	1060	12867	**434**	9/83	29.64
HMRKB Herath (SL)	1999-2018	93	170	25993	4332.1	814	12157	**433**	9/127	28.07
Sir RJ Hadlee (NZ)	1973-1990	86	150	21918	-	809	9611	**431**	9/52	22.29
SM Pollock (SA)	1995-2008	108	202	24353	4058.5	1222	9733	**421**	7/87	23.11
Harbhajan Singh (IND)	1998-2015	103	190	28580	4763.2	871	13537	**417**	8/84	32.46
Wasim Akram (PAK)	1985-2002	104	181	22627	3771.1	871	9779	**414**	7/119	23.62
CEL Ambrose (WI)	1988-2000	98	179	22103	3683.5	1001	8501	**405**	8/45	20.99
M Ntini (SA)	1998-2009	101	190	20834	3472.2	759	11242	**390**	7/37	28.82
IT Botham (ENG)	1977-1992	102	168	21815	-	788	10878	**383**	8/34	28.40
TG Southee (NZ)	2008-2024	100	190	22614	3769.0	868	11255	**380**	7/64	29.61
MD Marshall (WI)	1978-1991	81	151	17584	2930.4	614	7876	**376**	7/22	20.94
Waqar Younis (PAK)	1989-2003	87	154	16224	2704.0	516	8788	**373**	7/76	23.56
Imran Khan (PAK)	1971-1992	88	142	19458	-	727	8258	**362**	8/58	22.81
DL Vettori (ICC/NZ)	1997-2014	113	187	28814	4802.2	1197	12441	**362**	7/87	34.36
MA Starc (AUS)	2011-2024	89	170	17417	2902.5	559	9932	**358**	6/50	27.74
DK Lillee (AUS)	1971-1984	70	132	18467	-	652	8493	**355**	7/83	23.92
WPUJC Vaas (SL)	1994-2009	111	194	23438	3906.2	895	10501	**355**	7/71	29.58
AA Donald (SA)	1992-2002	72	129	15519	2586.3	661	7344	**330**	8/71	22.25
RGD Willis (ENG)	1971-1984	90	165	17357	-	554	8190	**325**	8/43	25.20
TA Boult (NZ)	2011-2022	78	149	17417	2902.5	656	8717	**317**	6/30	27.49
MG Johnson (AUS)	2007-2015	73	140	16001	2666.5	514	8891	**313**	8/61	28.40
I Sharma (IND)	2007-2021	105	188	19160	3193.2	640	10078	**311**	7/74	32.40
Z Khan (IND)	2000-2014	92	165	18785	3130.5	624	10247	**311**	7/87	32.94



untitled1.py Last Checkpoint: 24 seconds ago

dit View Settings Help

```
 Select features and target variable
 = data[['Inns', 'Balls', 'Runs', 'Wkts
Strike_Rate', 'Economy_Rate']]
 = data['Ave']

 Split data into training and testing s
rom sklearn.model_selection import
rain_test_split
_train, X_test, y_train, y_test =
rain_test_split(X, y, test_size=0.2)

 Train the linear regression model
odel = ols('Ave ~ Inns + Balls + Runs +
kts + Strike_Rate + Economy_Rate',
ata=X_train).fit()

 Make predictions on the testing set
_pred = model.predict(X_test)

 Evaluate the model's performance
rom sklearn.metrics import r2_score
2 = r2_score(y_test, y_pred)
rint('R-squared:', r2)

 Plot the actual vs predicted bowling
verage
mport matplotlib.pyplot as plt
lt.scatter(y_test, y_pred)
lt.xlabel('Actual Bowling Average')
lt.ylabel('Predicted Bowling Average')
lt.title('Linear Regression Model')
lt.show()
```



untitled1.py Last Checkpoint: 24 seconds ago

File Edit View Settings Help

```
# Select features and target variable
X = data[['Inns', 'Balls', 'Runs', 'Wkts',
'Strike_Rate', 'Economy_Rate']]
y = data['Ave']

# Split data into training and testing sets
from sklearn.model_selection import
train_test_split
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2)

# Train the linear regression model
model = ols('Ave ~ Inns + Balls + Runs +
Wkts + Strike_Rate + Economy_Rate',
data=X_train).fit()

# Make predictions on the testing set
y_pred = model.predict(X_test)

# Evaluate the model's performance
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print('R-squared:', r2)

# Plot the actual vs predicted bowling
average
import matplotlib.pyplot as plt
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Bowling Average')
plt.ylabel('Predicted Bowling Average')
plt.title('Linear Regression Model')
plt.show()
```

Thankyou