Multiple Linear Regression

Introduction

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_k X_{ki} + \varepsilon_i$$

partial regression coefficients

Assumptions

- Regression model is linear in regression parameters (b-values)
- residuals follow a normal distribution
  - the expected value (mean) of the residuals is zero
- In time series data, residuals are assumed to uncorrelated.
- The variance of the residuals is constant for all values of Xi.
- There is no high correlation between independent variables in the model.

Predicting the SOLD PRICE (Auction Price) of Players

Example Dataset: Predicting the SOLD PRICE (Auction Price) of IPL Players

- **Context:** The **Indian Premier League (IPL)** is a professional cricket league started in 2008, where franchises bid for players in an annual auction.
- **Auction Process:** Players are acquired through an **English auction system** where international and popular Indian players are bid upon.
- **Objective:** Predict the **SOLD PRICE (Auction Price)** of a player based on various performance metrics.
- **Data:** Performance records of **130 players** from IPL seasons **2008–2011**.
- **Features:**
  - Batting metrics (e.g., **batting average, strike rate**)
  - Bowling metrics (e.g., **wickets taken, economy rate**)
  - Fielding records and other performance indicators from **T20, ODI, and Test cricket**

How This Relates to Multiple Linear Regression

- **Dependent Variable (Target):** SOLD PRICE (Auction Price)
- **Independent Variables (Features):** Player performance statistics across different cricket formats
- **Goal:** Build a predictive model to estimate a player's value in future IPL auctions based on historical performance.

Dataset

TABLE 4.3 Metadata of IPL dataset

Data Code	Description
AGE	Age of the player at the time of auction classified into three categories. Category 1 (L25) means the player is less than 25 years old, category 2 means that the age is between 25 and 35 years (B25– 35) and category 3 means that the age is more than 35 (A35).
RUNS-S	Number of runs scored by a player.
RUNS-C	Number of runs conceded by a player.
HS	Highest score by a batsman in IPL.
AVE-B	Average runs scored by a batsman in IPL.
AVE-BL	Bowling average (number of runs conceded/number of wickets taken) in IPL.
SR-B	Batting strike rate (ratio of the number of runs scored to the number of balls faced) in IPL.
SR-BL	Bowling strike rate (ratio of the number of balls bowled to the number of wickets taken) in IPL.
SIXERS	Number of six runs scored by a player in IPL.
WKTS	Number of wickets taken by a player in IPL.

Data Code	Description
ECON	Economy rate of a bowler (number of runs conceded by the bowler per over) in IPL.
CAPTAINCY EXP	Captained either a T20 team or a national team.
ODI-SR-B	Batting strike rate in One-Day Internationals.
ODI-SR-BL	Bowling strike rate in One-Day Internationals.
ODI-RUNS-S	Runs scored in One-Day Internationals.
ODI-WKTS	Wickets taken in One-Day Internationals.
T-RUNS-S	Runs scored in Test matches.
T-WKTS	Wickets taken in Test matches.
PLAYER-SKILL	Player's primary skill (batsman, bowler, or allrounder).
COUNTRY	Country of origin of the player (AUS: Australia; IND: India; PAK: Pakistan; SA: South Africa; SL: Sri Lanka; NZ: New Zealand; WI: West Indies; OTH: Other countries).
YEAR-A	Year of Auction in IPL.
IPL TEAM	Team(s) for which the player had played in the IPL (CSK: Chennai Super Kings; DC: Deccan Chargers; DD: Delhi Dare-devils; KXI: Kings XI Punjab; KKR: Kolkata Knight Riders; MI: Mumbai Indians; PWI: Pune Warriors India; RR: Rajasthan Royals; RCB: Royal Challengers Bangalore). A + sign is used to indicate that the player has played for more than one team. For example, CSK+ would mean that the player has played for CSK as well as for one or more other teams.

Developing Multiple Linear Regression Model Using Python

4.5.2.1 Loading the Dataset

Loading data from *`IPL IMB381IPL2013.csv`* the file and print the meta data.

```
ipl_auction_df = pd.read_csv( 'IPL IMB381IPL2013.csv' )
```

```
ipl_auction_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Rangelndex: 130 entries, 0 to 129
Data columns (total 26 columns):
Sl.NO.              130    non-null   int64
PLAYER NAME         130    non-null   object
AGE                 130    non-null   int64
COUNTRY             130    non-null   object
TEAM                130    non-null   object
```

Dataset

Loading the dataset

```
PLAYING ROLE      130   non-null   object
T-RUNS            130   non-null   int64
T-WKTS            130   non-null   int64
ODI-RUNS-S        130   non-null   int64
ODI-SR-B          130   non-null   float64
ODI-WKTS          130   non-null   int64
ODI-SR-BL         130   non-null   float64
CAPTAINCY EXP     130   non-null   int64
RUNS-S            130   non-null   int64
HS                130   non-null   int64
AVE               130   non-null   float64
SR-B              130   non-null   float64
SIXERS            130   non-null   int64
RUNS-C            130   non-null   int64
WKTS              130   non-null   int64
AVE-BL            130   non-null   float64
ECON              130   non-null   float64
SR-BL             130   non-null   float64
AUCTION YEAR      130   non-null   int64
BASE PRICE        130   non-null   int64
SOLD PRICE        130   non-null   int64
dtypes: float64(7), int64(15), object(4)
memory usage: 26.5+ KB
```

Dataset

Displaying the First Five Records

```
ipl_auction_df.iloc[0:5, 0:10]
```

	SI. NO.	PLAYER NAME	AGE	COUNTRY	TEAM	PLAYING ROLE	T-RUNS	T-WKTS	ODI-RUNS-S	ODI-SR-B
0	1	Abdulla, YA	2	SA	KXIP	Allrounder	0	0	0	0.00
1	2	Abdur Razzak	2	BAN	RCB	Bowler	214	18	657	71.41
2	3	Agarkar, AB	2	IND	KKR	Bowler	571	58	1269	80.62
3	4	Ashwin, R	1	IND	CSK	Bowler	284	31	241	84.56
4	5	Badrinath, S	2	IND	CSK	Batsman	63	0	79	45.93

Dataset

Identify Relevant Features:

- Focus on player statistics that directly impact auction price.
- Ignore columns that do not contribute to prediction, such as serial numbers or timestamps.

Exclude Non-Feature Columns:

- Remove columns like **SI. NO.**, **BASE PRICE**, and any other irrelevant metadata.

Create a Feature List (`X_feature`):

- Define a list of features that should be used for modeling.
- Include only meaningful player statistics like runs, wickets, strike rate, economy rate, etc.

Filter DataFrame Based on Selected Features:

- Extract only the relevant columns from the dataset.
- Ensure consistency by handling missing values and data types.

Dataset

Identify Relevant Features:

```
X_features = ipl_auction_df.columns
```

```
X_features = ['AGE', 'COUNTRY', 'PLAYING ROLE',
              'T-RUNS', 'T-WKTS', 'ODI-RUNS-S', 'ODI-SR-B',
              'ODI-WKTS', 'ODI-SR-BL', 'CAPTAINCY EXP', 'RUNS-S',
              'HS', 'AVE', 'SR-B', 'SIXERS', 'RUNS-C', 'WKTS',
              'AVE-BL', 'ECON', 'SR-BL']
```

Dataset

Encoding Categorical Features

```
ipl_auction_df['PLAYING ROLE'].unique()
```

```
array(['Allrounder', 'Bowler', 'Batsman', 'W. Keeper'], dtype=object)
```

```
pd.get_dummies(ipl_auction_df['PLAYING ROLE'])[0:5]
```

	Allrounder	Batsman	Bowler	W. Keeper
0	1.0	0.0	0.0	0.0
1	0.0	0.0	1.0	0.0
2	0.0	0.0	1.0	0.0
3	0.0	0.0	1.0	0.0
4	0.0	1.0	0.0	0.0

Dataset

Encoding Categorical Variable

```
ipl_auction_encoded_df.columns
```

```
Index(['T-RUNS', 'T-WKTS', 'ODI-RUNS-S', 'ODI-SR-B', 'ODI-WKTS',
       'ODI-SR-BL', 'RUNS-S', 'HS', 'AVE', 'SR-B', 'SIXERS',
       'RUNS-C', 'WKTS', 'AVE-BL', 'ECON', 'SR-BL', 'AGE_2',
       'AGE_3', 'COUNTRY_BAN', 'COUNTRY_ENG', 'COUNTRY_IND',
       'COUNTRY_NZ', 'COUNTRY_PAK', 'COUNTRY_SA', 'COUNTRY_SL',
       'COUNTRY_WI', 'COUNTRY_ZIM', 'PLAYING ROLE_Batsman',
       'PLAYING ROLE_Bowler', 'PLAYING ROLE_W. Keeper',
       'CAPTAINCY EXP_1'], dtype='object')
```

```
X_features = ipl_auction_encoded_df.columns
```

Dataset

Splitting the Dataset into Train and Validation Sets

python

Copy Edit

```python
from sklearn.model_selection import train_test_split

# Define features (X) and target variable (Y)
X = ipl_auction_encoded_df  # Player statistics (processed features)
Y = ipl_auction_df['SOLD PRICE']  # Target variable

# Split data into training (80%) and testing (20%) sets
train_X, test_X, train_y, test_y = train_test_split(X, Y, train_size=0.8, random_stat

# Print dataset sizes
print("Training data shape:", train_X.shape, train_y.shape)
print("Testing data shape:", test_X.shape, test_y.shape)
```

Model

Building the Model on the Training Dataset

```
ipl_model_1 = sm.OLS(train_y, train_X).fit()
ipl_model_1.summary2()
```

TABLE 4.4 Model summary for *ipl_model_1*

Model:	OLS	Adj. R-squared:	0.362
Dependent Variable:	SOLD PRICE	AIC:	2965.2841
Date:	2018-04-08 07:27	BIC:	3049.9046
No. Observations:	104	Log-Likelihood:	–1450.6
Df Model:	31	F-statistic:	2.883
Df Residuals:	72	Prob (F-statistic):	0.000114
R-squared:	0.554	Scale:	1.1034e+11

Model

	Coef.	Std.Err.	t	P > |t|	[0.025	0.975]
const	375827.1991	228849.9306	1.6422	0.1049	−80376.7996	832031.1978
T-RUNS	−53.7890	32.7172	−1.6441	0.1045	−119.0096	11.4316
T-WKTS	−132.5967	609.7525	−0.2175	0.8285	−1348.1162	1082.9228
ODI-RUNS-S	57.9600	31.5071	1.8396	0.0700	−4.8482	120.7681
ODI-SR-B	−524.1450	1576.6368	−0.3324	0.7405	−3667.1130	2618.8231
ODI-WKTS	815.3944	832.3883	0.9796	0.3306	−843.9413	2474.7301
ODI-SR-BL	−773.3092	1536.3334	−0.5033	0.6163	−3835.9338	2289.3154
RUNS-S	114.7205	173.3088	0.6619	0.5101	−230.7643	460.2054
HS	−5516.3354	2586.3277	−2.1329	0.0363	−10672.0855	−360.5853
AVE	21560.2760	7774.2419	2.7733	0.0071	6062.6080	37057.9439

(*Continued*)

	Coef.	Std.Err.	t	P > |t|	[0.025	0.975]
SR-B	−1324.7218	1373.1303	−0.9647	0.3379	−4062.0071	1412.5635
SIXERS	4264.1001	4089.6000	1.0427	0.3006	−3888.3685	12416.5687
RUNS-C	69.8250	297.6697	0.2346	0.8152	−523.5687	663.2187
WKTS	3075.2422	7262.4452	0.4234	0.6732	−11402.1778	17552.6622
AVE-BL	5182.9335	10230.1581	0.5066	0.6140	−15210.5140	25576.3810
ECON	−6820.7781	13109.3693	−0.5203	0.6045	−32953.8282	19312.2721
SR-BL	−7658.8094	14041.8735	−0.5454	0.5871	−35650.7726	20333.1539
AGE_2	−230767.6463	114117.2005	−2.0222	0.0469	−458256.1279	−3279.1648
AGE_3	−216827.0808	152246.6232	−1.4242	0.1587	−520325.1772	86671.0155
COUNTRY_BAN	−122103.5196	438719.2796	−0.2783	0.7816	−996674.4194	752467.3801
COUNTRY_ENG	672410.7654	238386.2220	2.8207	0.0062	197196.5172	1147625.0135
COUNTRY_IND	155306.4011	126316.3449	1.2295	0.2229	−96500.6302	407113.4325
COUNTRY_NZ	194218.9120	173491.9293	1.1195	0.2667	−151630.9280	540068.7521
COUNTRY_PAK	75921.7670	193463.5545	0.3924	0.6959	−309740.7804	461584.3143
COUNTRY_SA	64283.3894	144587.6773	0.4446	0.6579	−223946.8775	352513.6563
COUNTRY_SL	17360.1530	176333.7497	0.0985	0.9218	−334154.7526	368875.0586
COUNTRY_WI	10607.7792	230686.7892	0.0460	0.9635	−449257.9303	470473.4887
COUNTRY_ZIM	−145494.4793	401505.2815	−0.3624	0.7181	−945880.6296	654891.6710
PLAYING ROLE_Batsman	75724.7643	150250.0240	0.5040	0.6158	−223793.1844	375242.7130
PLAYING ROLE_Bowler	15395.8752	126308.1272	0.1219	0.9033	−236394.7744	267186.5249
PLAYING ROLE_W. Keeper	−71358.6280	213585.7444	−0.3341	0.7393	−497134.0278	354416.7718
CAPTAINCY EXP_1	164113.3972	123430.6353	1.3296	0.1878	−81941.0772	410167.8716

Model

📌 **Key Findings from the Model**

- Only **HS, AGE_2, AVE, and COUNTRY_ENG** are **statistically significant** (p-value < 0.05).
- Other features appear **insignificant** in predicting SOLD PRICE.

📌 **Why Does This Happen?**

- **Multi-collinearity**: Some features are **highly correlated**, making their individual impact unclear.
- The model distributes effects among correlated variables, increasing **p-values** for some.

📌 **How to Check Multi-Collinearity?**

- **Variance Inflation Factor (VIF)**: Detects redundant features.
- **Correlation Matrix**: Identifies highly correlated features.

p-value

What is p-value?

The **p-value** (probability value) is a statistical measure that helps determine whether an observed effect in data is **statistically significant** or just **due to random chance**.

Intuition Behind p-value

Imagine you are testing whether a player's **batting average (AVE)** has an impact on their **auction price (SOLD PRICE)** in a regression model.

- **Null Hypothesis (H0H_0H0)**: The player's **batting average has no effect** on SOLD PRICE.
- **Alternative Hypothesis (H1H_1H1)**: The player's **batting average does influence** SOLD PRICE.

The **p-value** tells us **how likely** we would get the observed effect **if the null hypothesis were true**.

p-value

Interpreting p-value

- **p-value < 0.05** (typically 5% significance level)
  → **Reject the null hypothesis** → The feature is statistically significant.
  → The feature (e.g., **batting average**) **has an impact** on SOLD PRICE.
- **p-value > 0.05**
  → **Fail to reject the null hypothesis** → The feature is not statistically significant.
  → The feature does **not** significantly influence SOLD PRICE.

Example in Regression

If you run a regression model to predict SOLD PRICE based on multiple features, you might get:

Feature	Coefficient	p-value
Batting Average (AVE)	12000	**0.02** ✅
Strike Rate (SR)	5000	**0.08** ❌
Age (AGE_2)	-3000	**0.01** ✅

- Since **AVE and AGE_2** have **p-values < 0.05**, they **significantly influence** SOLD PRICE.
- **Strike Rate (SR)** has a **p-value > 0.05**, meaning it **may not significantly affect** SOLD PRICE.

Multi-Collinearity and Handling Multi-Collinearity

📌 **How Multi-Collinearity Affects the Model?**

1️⃣ **Inflates Standard Error (Se(b)S_e(b)Se(b))**

- Makes coefficient estimates **less reliable**.

2️⃣ **Statistically Significant Features May Appear Insignificant**

- Large **p-value** due to high standard error.
- Leads to **underestimation** of t-statistic.

3️⃣ **Changes the Sign of Regression Coefficients**

- A **negative coefficient** may appear **positive** and vice versa.

4️⃣ **Regression Coefficients Become Unstable**

- **Adding or removing** a variable/observation **dramatically changes** estimates.

Multi-Collinearity and Handling Multi-Collinearity

Variance Inflation Factor (VIF)

📌 **What is VIF?**

- **Variance Inflation Factor (VIF)** is used to **detect multi-collinearity** between independent variables.

📌 **How is VIF Calculated?**

- Consider two independent variables X_1 **and** X_2:

$$X_1 = a_0 + a_1 X_2$$

- If R_{12}^2 is the **R-squared value**, then:

$$VIF = \frac{1}{1 - R_{12}^2}$$

- **VIF > 4** suggests strong multi-collinearity and requires further investigation.

Multi-Collinearity and Handling Multi-Collinearity

```
from statsmodels.stats.outliers_influence import variance_inflation_
factor

def get_vif_factors( X ):
        X_matrix = X.as_matrix()
        vif = [ variance_inflation_factor( X_matrix, i ) for i in range
              ( X_matrix.sh ape[1] ) ]
        vif_factors = pd.DataFrame()
        vif_factors['column'] = X.columns
        vif_factors['VIF'] = vif

        return vif_factors
```

```
vif_factors = get_vif_factors( X[X_features] )
vif_factors
```

Multi-Collinearity and Handling Multi-Collinearity

	Column	VIF
1	T-WKTS	7.679284
2	ODI-RUNS-S	16.426209
3	ODI-SR-B	13.829376
4	ODI-WKTS	9.951800
5	ODI-SR-BL	4.426818
6	RUNS-S	16.135407
7	HS	22.781017
8	AVE	25.226566
9	SR-B	21.576204
10	SIXERS	9.547268
11	RUNS-C	38.229691
12	WKTS	33.366067
13	AVE-BL	100.198105
14	ECON	7.650140
15	SR-BL	103.723846
16	AGE_2	6.996226
17	AGE_3	3.855003
18	COUNTRY_BAN	1.469017
19	COUNTRY_ENG	1.391524
20	COUNTRY_IND	4.568898
21	COUNTRY_NZ	1.497856
22	COUNTRY_PAK	1.796355
23	COUNTRY_SA	1.886555
24	COUNTRY_SL	1.984902
25	COUNTRY_WI	1.531847
26	COUNTRY_ZIM	1.312168
27	PLAYING ROLE_Batsman	4.843136
28	PLAYING ROLE_Bowler	3.795864
29	PLAYING ROLE_W. Keeper	3.132044
30	CAPTAINCY EXP_1	4.245128

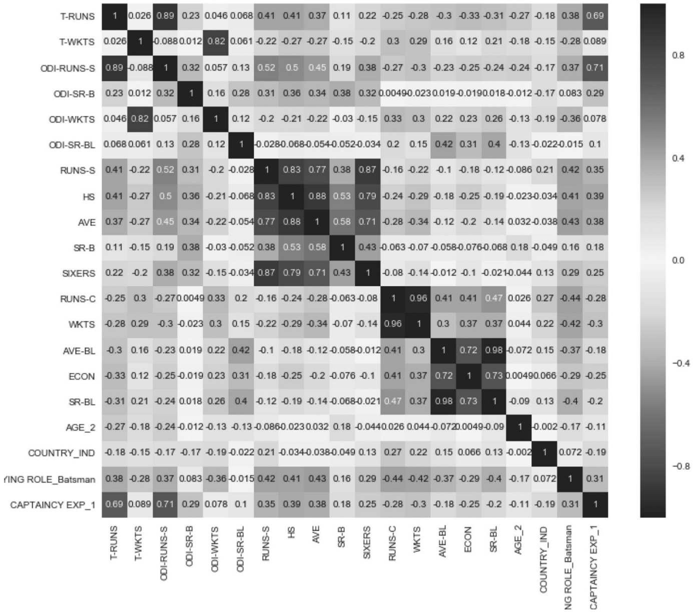
Multi-Collinearity and Handling Multi-Collinearity

Checking Correlation of Columns with Large VIFs

```
columns_with_large_vif = vif_factors[vif_factors.vif > 4].column
```

```
plt.figure( figsize = (12,10) )
sn.heatmap( X[columns_with_large_vif].corr(), annot = True );
plt.title("Figure 4.5 - Heatmap depicting correlation between
features");
```

Multi-Collinearity and Handling Multi-Collinearity



Multi-Collinearity and Handling Multi-Collinearity

📌 **Key Correlations Identified**

1️⃣ **Strong Correlation Between Batting & Bowling Stats**

- **T-RUNS** and **ODI-RUNS-S** are highly correlated.
- **ODI-WKTS** and **T-WKTS** are also highly correlated.

2️⃣ **Batsman-Specific Correlations**

- **RUNS-S, HS, AVE, SIXERS** are strongly correlated.

3️⃣ **Bowler-Specific Correlations**

- **AVE-BL, ECON, and SR-BL** show high correlation.

📌 **Why Does This Matter?**

- **Multi-collinearity** can affect regression models.
- **Redundant features** may reduce model interpretability.

Multi-Collinearity and Handling Multi-Collinearity

Feature Selection

📌 **Why Remove Features?**

- Highly correlated features can lead to **multi-collinearity**, making model predictions unstable.
- Keeping redundant features **inflates standard errors** and affects statistical significance.

📌 **How to Decide Which Features to Keep?**
✅ **Domain Knowledge**: Understand which features best represent player performance.
✅ **Statistical Analysis**: Use **VIF (Variance Inflation Factor)** and **correlation heatmaps**.
✅ **Model Performance**: Test different feature sets in **multiple iterations**.

📌 **Feature Removal Strategy**

- Identify **highly correlated** variables.
- Remove **one from each correlated pair** based on interpretability.
- Iterate and refine the feature set **until multi-collinearity is minimized**.

🚀 **Final Features Selected After Iterations**
✔ **Retained Features:** Key statistics with minimal collinearity.
❌ **Removed Features:** Redundant or highly correlated variables.

```
columns_to_be_removed = ['T-RUNS', 'T-WKTS', 'RUNS-S', 'HS', 'AVE',
                         'RUNS-C', 'SR-B', 'AVE-BL', 'ECON',
                         'ODI-SR-B', 'ODI-RUNS-S', 'AGE_2', 'SR-BL' ]
```

```
X_new_features = list( set(X_features) - set(columns_to_be_removed))
```

```
get_vif_factors( X[X_new_features] )
```

	Column	VIF
0	AGE_3	1.779861
1	ODI-SR-BL	2.822148
2	COUNTRY_IND	3.144668
3	COUNTRY_ENG	1.131869
4	COUNTRY_NZ	1.173418
5	COUNTRY_PAK	1.334773
6	COUNTRY_WI	1.194093
7	COUNTRY_SL	1.519752
8	COUNTRY_ZIM	1.205305
9	CAPTAINCY EXP_1	2.458745
10	PLAYING ROLE_W. Keeper	1.900941
11	PLAYING ROLE_Bowler	3.060168
12	SIXERS	2.397409
13	COUNTRY_BAN	1.094293
14	COUNTRY_SA	1.416657
15	PLAYING ROLE_Batsman	2.680207
16	ODI-WKTS	2.742889
17	WKTS	2.883101

Building a New Model after Removing Multi-collinearity

```
train_X = train_X[X_new_features]

ipl_model_2 = sm.OLS(train_y, train_X).fit()
ipl_model_2.summary2()
```

TABLE 4.5 Model summary for *ipl_model_2*

Model:	OLS	Adj. R-squared:	0.728
Dependent Variable:	SOLD PRICE	AIC:	2965.1080
Date:	2018-04-08 07:27	BIC:	3012.7070
No. Observations:	104	Log-Likelihood:	–1464.6
Df Model:	18	F-statistic:	16.49
Df Residuals:	86	Prob (F-statistic):	1.13e-20
R-squared:	0.775	Scale:	1.2071e+11

Feature Selection using p-values

📌 **Understanding p-values**

- A feature is **statistically significant** if **p-value < 0.05** (or chosen significance level **α\alphaα**).
- **Higher p-values** indicate features **do not significantly** impact SOLD PRICE.

📌 **Statistically Significant Features (p-value < 0.05)**
✔ **COUNTRY_IND, COUNTRY_ENG** → Player's nationality matters.
✔ **SIXERS** → Number of sixes hit in past IPL seasons.
✔ **CAPTAINCY_EXP_1** → Previous captaincy experience.

📌 **Key Insights from the Model**
1️⃣ **Player's Origin Matters**

- Players from **India** and **England** significantly influence SOLD PRICE.

2️⃣ **Performance in Previous IPLs & ODIs**

- Number of **sixes hit** and **wickets taken in ODIs** impact pricing.

3️⃣ **Leadership Experience**

- **Captaincy history** plays a crucial role in auction pricing.

	Coef.	Std.Err.	t	$P>\|t\|$	[0.025	0.975]
COUNTRY_IND	282829.8091	96188.0292	2.9404	0.0042	91614.3356	474045.2827
COUNTRY_BAN	−108758.6040	369274.1916	−0.2945	0.7691	−842851.4010	625334.1930
AGE_3	−8950.6659	98041.9325	−0.0913	0.9275	−203851.5772	185950.2453
COUNTRY_PAK	122810.2480	159600.8063	0.7695	0.4437	−194465.6541	440086.1502
COUNTRY_WI	−22234.9315	213050.5847	−0.1044	0.9171	−445765.4766	401295.6135
ODI-WKTS	772.4088	470.6354	1.6412	0.1044	−163.1834	1708.0009
COUNTRY_SA	108735.9086	115092.9596	0.9448	0.3474	−120061.3227	337533.1399
COUNTRY_ENG	682934.7166	216150.8279	3.1595	0.0022	253241.0920	1112628.3411
CAPTAINCY EXP_1	208376.6957	98128.0284	2.1235	0.0366	13304.6315	403448.7600
WKTS	2431.8988	2105.3524	1.1551	0.2512	−1753.4033	6617.2008
SIXERS	7862.1259	2086.6101	3.7679	0.0003	3714.0824	12010.1694
PLAYING ROLE_W. Keeper	−55121.9240	169922.5271	−0.3244	0.7464	−392916.7280	282672.8801
COUNTRY_ZIM	−67977.6781	390859.9289	−0.1739	0.8623	−844981.5006	709026.1444
PLAYING ROLE_Bowler	−18315.4968	106035.9664	−0.1727	0.8633	−229108.0215	192477.0279
COUNTRY_SL	55912.3398	142277.1829	0.3930	0.6953	−226925.3388	338750.0184
COUNTRY_NZ	142968.8843	151841.7382	0.9416	0.3491	−158882.5009	444820.2695
PLAYING ROLE_Batsman	121382.0570	106685.0356	1.1378	0.2584	−90700.7746	333464.8886
ODI-SR-BL	909.0021	1267.4969	0.7172	0.4752	−1610.6983	3428.7026

📌 Key Inferences from the Latest Model (`ipl_model_3`)

1️⃣ All Variables Are Statistically Significant

- **p-values < 0.05**, meaning all selected features influence **SOLD PRIC**

2️⃣ Overall Model Significance

- **F-statistic p-value < 0.05**, confirming the model is statistically valid.

3️⃣ Model Performance (R-squared & Adjusted R-squared)

- **R-squared = 0.715** → Model explains **71.5% variance** in SOL
- **Adjusted R-squared = 0.704** → Accounts for the number of p

Let us create a new list called *significant_vars* to store the column names of significant variables and build a new model (Table 4.6).

```
significant_vars = ['COUNTRY_IND', 'COUNTRY_ENG', 'SIXERS',
'CAPTAINCY EXP_1']
train_X = train_X[significant_vars]
ipl_model_3 = sm.OLS(train_y, train_X).fit()
ipl_model_3.summary2()
```

	Coef.	Std.Err.	t	$P > \|t\|$	[0.025	0.975]
COUNTRY_IND	387890.2538	63007.1511	6.1563	0.0000	262885.8606	512894.6471
COUNTRY_ENG	731833.6386	214164.4988	3.4172	0.0009	306937.3727	1156729.9045
SIXERS	8637.8344	1675.1313	5.1565	0.0000	5314.4216	11961.2472
CAPTAINCY EXP_1	359725.2741	74930.3460	4.8008	0.0000	211065.6018	508384.9463

📌 Why Adjusted R-squared?

✅ Normalizes **SSE (Sum of Squared Errors)** and **SST (Total Sum of Squares)**.
✅ Penalizes unnecessary features to ensure a **better fit**.