# Machine Learning-Session 3

## AI Labs

Gourav Bansal

# Machine Learning Session Three

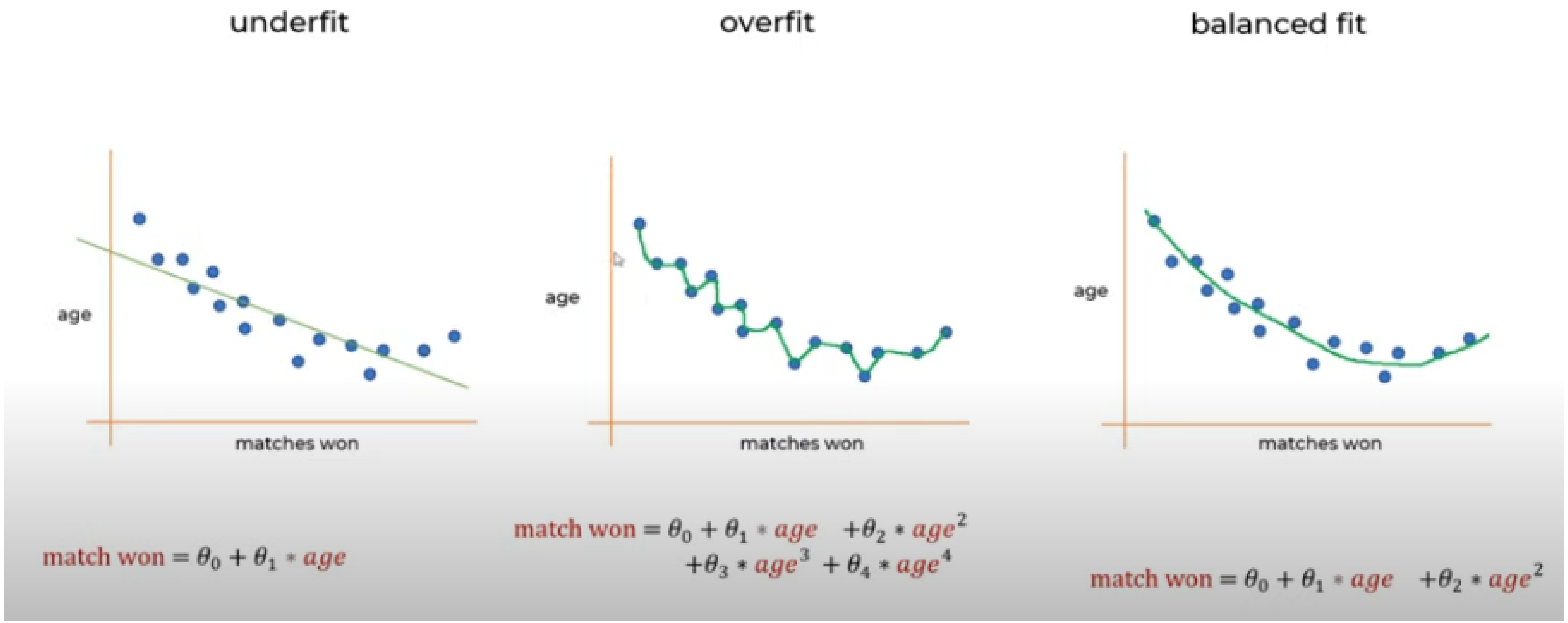
## Agenda

- Penalty
- Lasso Regression
- Ridge Regression
- Mean Absolute Error
- Mean Squared Error
- Root Mean Square Error
- Root Mean Square Logarithmic Error
- R2-Score

# Penalty ?

A penalty, also known as regularization, is a technique used to prevent overfitting by adding a penalty term to the model's loss function. This penalty discourages the model from learning overly complex patterns from the training data that might not generalize well to new, unseen data. Essentially, it forces the model to find a balance between fitting the training data well and keeping its parameters relatively small or simple.

# Fits

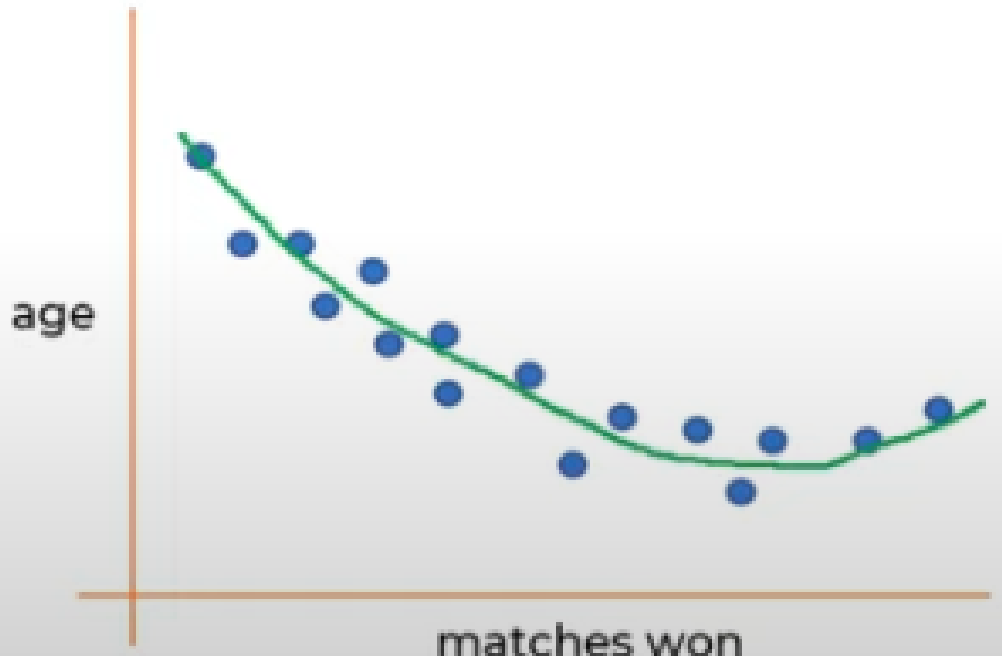

underfit
overfit
balanced fit
age
age
age
matches won
matches won
matches won
$\text{match won} = \theta_0 + \theta_1 * age$
$\text{match won} = \theta_0 + \theta_1 * age \quad + \theta_2 * age^2 + \theta_3 * age^3 + \theta_4 * age^4$
$\text{match won} = \theta_0 + \theta_1 * age \quad + \theta_2 * age^2$

# Continue...


$$\text{match won} = \theta_0 + \theta_1 * age \; + \theta_2 * age^2 + \theta_3 * age^3 + \theta_4 * age^4$$

Try to make $\theta_3$ and $\theta_4$ almost close to zero

$$\text{match won} = \theta_0 + \theta_1 * age \; + \theta_2 * age^2$$



$$\text{Loss}_{\text{regularized}} = \text{MSE} + \lambda \cdot \text{Penalty}$$

Where:

- $\lambda$ = regularization strength (hyperparameter)
- Penalty = function of model weights (like L1 or L2)

| `lambda` value | Meaning | Effect on model |
| --- | --- | --- |
| **Very small** (e.g., 0 or 1e-4) | Almost no penalty | May **overfit** |
| **Medium** (e.g., 0.1 to 10) | Balanced penalty | Good generalization |
| **Large** (e.g., 100 or more) | Heavy penalty on coefficients | May **underfit** or push weights to zero |

# Mean Squared Error

$$mse = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_{predicted}\right)^2$$

## Mean Squared Error

$$ms\varepsilon = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_\theta(x_i)\right)^2$$

$$h_\theta(x_i) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$$

Ridge Regression

## L2 Regularization

$$ms\varepsilon = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_\theta(x_i)\right)^2 + \lambda \sum_{i=1}^{n} \theta_i^2$$

$$h_\theta(x_i) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$$

Lasso Regression

## L1 Regularization

$$mse = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_\theta(x_i)\right)^2 + \lambda \sum_{i=1}^{n}|\theta_i|$$

$$h_\theta(x_i) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$$

# Mean Absolute Error

Divide by the total number of data points

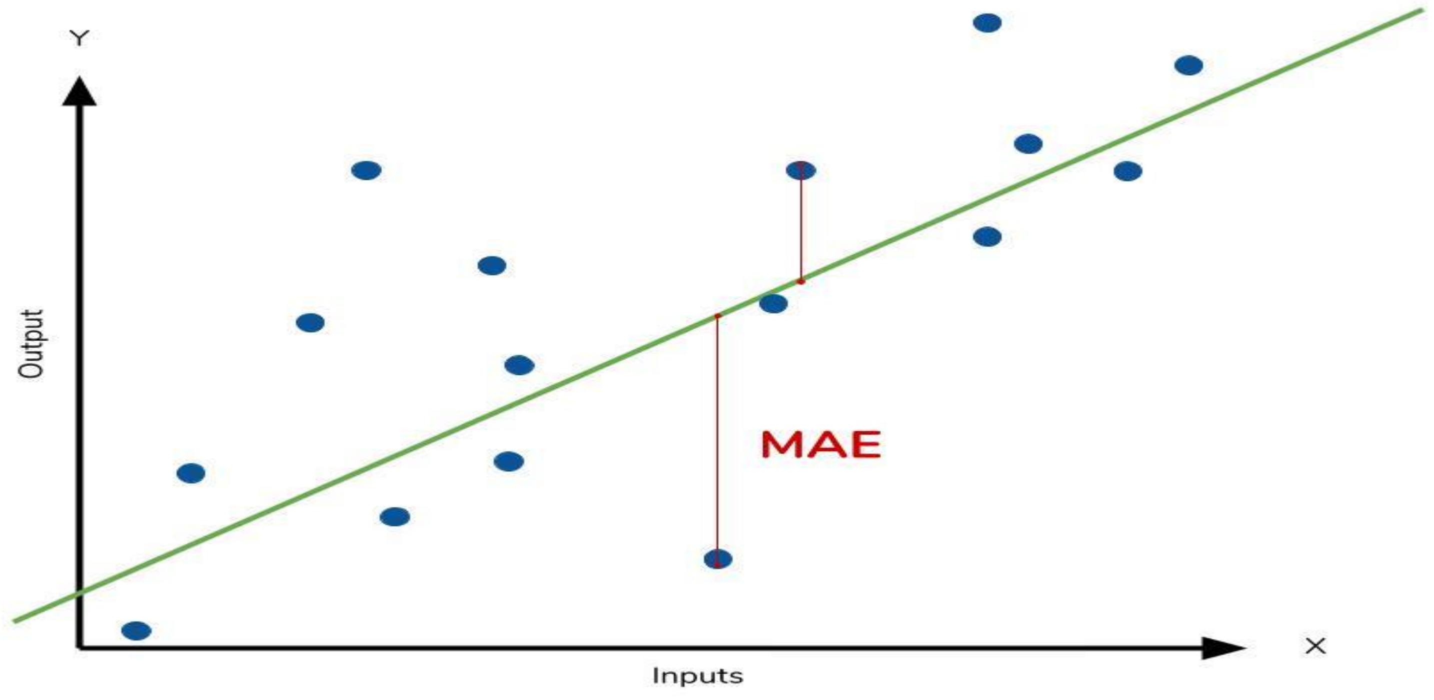
Predicted output value

Actual output value

$$MAE = \boxed{\frac{1}{n}} \sum \underbrace{\left| \boxed{y} - \boxed{\hat{y}} \right|}_{\text{The absolute value of the residual}}$$

Sum of

# Mean Absolute Error




### ✅ When to Use MAE

- You want a metric that is **easy to interpret** (in same units as the target).
- You want to treat all errors **equally** (no squaring).
- Your data has **outliers** and you don't want them to dominate the error.

---

### 🚫 When Not Ideal

- Doesn't penalize large errors as strongly as MSE or RMSE.
- Not differentiable at 0 (which can matter for some optimization algorithms).

---

### 📘 Example

| **Actual** $y$ | **Predicted** $\hat{y}$ | **Error** $\hat{y} - y$ | **Absolute Error** |
|---|---|---|---|
| 100 | 90 | -10 | 10 |
| 200 | 220 | +20 | 20 |
| 300 | 290 | -10 | 10 |

$$\mathrm{MAE} = \frac{10 + 20 + 10}{3} = \frac{40}{3} \approx 13.33$$

# Mean Squared Error

$$\mathrm{MSE} = \overset{\text{Mean}}{\boxed{\frac{1}{n}\sum_{i=1}^{n}}}\Big(\overset{\text{Error}}{\boxed{Y_i - \hat{Y}_i}}\Big)^{\overset{\text{Squared}}{\boxed{2}}}$$

$$\underbrace{\sum_{i=1}^{\overset{\text{number of samples}}{\boxed{n}}}\Big(\overset{\text{real value}}{\boxed{Y_i}} - \overset{\text{predicted value}}{\boxed{\hat{Y}_i}}\Big)^2}_{\text{sum of the errors of all samples}}$$

### ✅ When to Use MSE

- You want to **penalize larger errors** more than smaller ones.
- You're comparing models and don't need the error in original units (unlike RMSE).
- You're optimizing a regression algorithm (many models use MSE as a loss function).

### 🚫 When Not Ideal

- Not interpretable in original units (e.g., if predicting prices in ₹, MSE is in ₹²).
- Sensitive to **outliers**, as squaring amplifies large errors.

### 📘 Example

| Actual $y$ | Predicted $\hat{y}$ | Error $\hat{y} - y$ | Squared Error |
|---|---|---|---|
| 100 | 90 | -10 | 100 |
| 200 | 220 | +20 | 400 |
| 300 | 290 | -10 | 100 |

$$\mathrm{MSE} = \frac{100 + 400 + 100}{3} = \frac{600}{3} = 200$$

# Root Mean Square Error

**Average of error**
The amout of error shouldn't be dependent on the range of x

**Square root of error**
Cancels the side-effect of applying the square

**Square of error**
By applying suqare, negative errors become positive

$$RMS = \sqrt{\boxed{\frac{1}{n}\sum_i}\boxed{(f_i - g_i)^2}}$$

= np.sqrt( np.sum((f - g)**2 ))/ n )

* f and g at the second line are 1-D numpy array, or pandas Series.
* We can use numpy's "mean()" function instead
* 1 over RMS turns to a measurement of how much f and g are close

### ✅ When to Use RMSE

- You want to **penalize large errors** more than small ones (due to squaring).
- You want to **measure model performance in original units** of the target variable.
- Suitable when the cost of large errors is high.

### 📘 Example

| Actual $y$ | Predicted $\hat{y}$ | Error $\hat{y} - y$ | Squared Error |
|---|---|---|---|
| 100 | 90 | -10 | 100 |
| 200 | 220 | +20 | 400 |
| 300 | 290 | -10 | 100 |

$$\mathrm{RMSE} = \sqrt{\frac{100 + 400 + 100}{3}} = \sqrt{200} \approx 14.14$$

# Root Mean Square Logarithmic Error

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\log_e(\hat{y}_i + 1) - \log_e(y_i + 1)\right)^2}$$

- $\hat{y}_i$: Predicted value
- $y_i$: Actual value
- $n$: Number of data points
- $\log_e$: Natural logarithm
- `+1` : To handle zero values in $y$ and $\hat{y}$

## ✅ When to Use RMSLE

- You're predicting **count data**, like number of clicks, views, or sales.
- You want **less penalty on large absolute errors for large values.**
- You want to **treat small differences in small values as significant.**

## 🚫 When Not to Use

- When actual and predicted values can be **negative** (log undefined).
- When absolute difference matters more than the ratio.

# R2-Score

**R² Score** measures how well your **regression model** explains the **variability** of the target variable. It gives an idea of how close the predicted values are to the actual values.

### 🔢 R² Score Formula

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- $y_i$: Actual value
- $\hat{y}_i$: Predicted value
- $\bar{y}$: Mean of actual values
- $n$: Number of samples

---

### 🎯 What R² Represents

- The proportion of variance in the dependent variable that's predictable from the independent variables.
- **R² = 1** → Perfect predictions
- **R² = 0** → Predictions are no better than just using the mean
- **R² < 0** → Model is **worse** than predicting the mean!

### ✅ When to Use

- To assess **goodness of fit** for regression.
- To compare how well different models perform on the **same dataset**.

---

### 📘 Example

| Actual $y$ | Predicted $\hat{y}$ | Mean of Actual $\bar{y}$ = 200 |
|---|---|---|
| 100 | 90 | |
| 200 | 220 | |
| 300 | 290 | |

Compute:

- **SSR** = $\sum(y - \hat{y})^2$ = (100-90)² + (200-220)² + (300-290)² = 100 + 400 + 100 = 600
- **TSS** = $\sum(y - \bar{y})^2$ = (100-200)² + (200-200)² + (300-200)² = 10000 + 0 + 10000 = 20000

$$R^2 = 1 - \frac{600}{20000} = 0.97$$

# Cheat Sheet

| Metric | What It Measures | ✅ Use When | ❌ Avoid When | 💡 Key Notes |
|---|---|---|---|---|
| **MAE** (Mean Absolute Error) | Average of absolute errors | You want a simple, interpretable error in the same unit as the target | You need to penalize large errors heavily | Treats all errors equally, not sensitive to outliers |
| **MSE** (Mean Squared Error) | Average of squared errors | You want to penalize large errors more; training models that optimize MSE | Interpretability is important; data has outliers | Very sensitive to large errors; unit is squared |
| **RMSE** (Root Mean Squared Error) | Square root of MSE (same units as target) | You want to penalize large errors and keep units readable | You want robustness against outliers | Balances interpretability with error severity |
| **RMSLE** (Root Mean Squared Log Error) | Log-scaled version of RMSE; measures relative error | Target has wide range or exponential growth (e.g., views, prices); relative error matters | When data contains negatives or many zeros | Penalizes under-prediction more than over; good for count data |
| **$R^2$ Score** (Coefficient of Determination) | Proportion of variance explained by the model | Assessing how well a model explains the target variable's variation | Comparing models across **different datasets** or if data has **low variance** | Can be negative; only meaningful with consistent target distribution |

# Comparision

# Usage Guide

- Use **MAE** when every error matters equally (robust & interpretable).
- Use **MSE/RMSE** when large errors need stronger penalty (common in model training).
- Use **RMSLE** for **relative prediction** and **log-distributed data** (e.g., sales, popularity).
- Use **$R^2$** to judge **overall model fit** — but don't rely on it alone.

# Notebook

Kaggle :

Linear Regression Collection
https://www.kaggle.com/work/collections/16193855

GitHub:

https://github.com/Ohanvi/machine-learning-module

Competition:

https://www.kaggle.com/competitions/predict-the-closing-stock-price

# Donate to India Army

- [Indian Army]
- [NDF - National Defense Fund]


| | | |
|---|---|---|
| **(a)** | **Name of Fund** | **: Army Central Welfare Fund.** |
| | Bank Name | : Union Bank of India |
| | Branch | : Chandni Chowk, Delhi – 110006 |
| | IFSC Code | : UBIN0530778 |
| | Account No | : 520101236373338 |
| | Type of Acct | : Saving |
| **(b)** | **Name of Fund** | **: Armed Forces Battle Casualties Welfare Fund.** |
| | Bank Name | : Canara Bank, |
| | Branch | : South Block, Defence Headquarters, New Delhi - 110011 |
| | IFSC Code | : CNRB0019055 |
| | Account No | : 90552010165915 |
| | Type of Acct | : Saving |

The End