Class #22: Machine Learning Fundamentals

1. Introduction to machine learning: Definition and importance.
2. Applications of machine learning.
3. Understanding supervised learning.
4. Introduction to unsupervised learning.
5. Overview of machine learning models.
6. Data splitting: Training and test sets.
7. Understanding K-fold cross-validation.
8. Handling underfitting and overfitting.
9. Confusion matrix metrics: Precision, recall, and F1 score.

Assignment #22:

1. Introduction to machine learning: Definition and importance.

Definition: Machine Learning হলো এমন একটি technology, যার মাধ্যমে computer বা system data থেকে শিখে এবং experience অর্জন করে, যাতে তারা future এ ভালো সিদ্ধান্ত নিতে পারে। Basically, এটি computer কে এমনভাবে train করে যাতে তারা automatically নির্দিষ্ট কাজ করতে পারে।

Importance: Machine Learning আমাদের everyday life এ অনেক কাজে আসে, যেমন:

- Image recognition
- Language translation
- Self-driving cars
- Product recommendation (যেমন, Amazon বা Netflix এ)

এটি বিভিন্ন fields যেমন business, healthcare, education, এবং financial services এ revolutionizing করছে এবং মানুষের কাজ অনেক সহজ করে দিচ্ছে।

Model:

Machine Learning এ **model** হচ্ছে মানুষের brain এর মতো, যা data থেকে শিখে সিদ্ধান্ত নেয়। আমাদের brain যেমন বিভিন্ন experiences থেকে শিখে future এ ভালো সিদ্ধান্ত নেয়, তেমনি modelও data দেখে patterns শিখে prediction বা decision নিতে পারে।

Model এর Accuracy যাচাই:

Model কতটুকু সত্য বা মিথ্যা বলছে, তা যাচাই করার জন্য **evaluation** করতে হয়। যেমন, **testing data** দিয়ে model এর prediction এর accuracy চেক করা হয়।

Model as Statistical Algorithm:

Machine Learning এর model মূলত **statistical algorithms** এর উপর কাজ করে। Statistical formula গুলি machine learning এর model হিসেবে কম্পিউটারে ব্যবহার করা হয়। উদাহরণ হিসেবে, **Linear Regression** একটি জনপ্রিয় model, যা data points এর মধ্যে relationship বুঝে future value predict করে।

উদাহরণ: ধরা যাক, আমরা একটি **model** তৈরি করেছি যাতে **house price** predict করা যায়। এখানে model গুলি past house prices থেকে শিখে price predict করবে। এখন, যদি model ১০টি বাড়ির দাম predict করে এবং তার মধ্যে ৮টি সঠিক হয়, তবে accuracy ভালো এবং model এর performance ভালো। কিন্তু, যদি model ভুলভাবে ১০টি prediction করে, তাহলে আমাদের model কে **improve** করতে হবে।

এইভাবে, model পরিসংখ্যান এর সূত্র এবং কম্পিউটার এর সাহায্যে আমাদের data থেকে শিখে কাজ করে।

2. Applications of machine learning.

Machine Learning এর applications বলতে বুঝায়, এটি কোথায় এবং কীভাবে ব্যবহার করা হচ্ছে। Machine Learning আজকাল বিভিন্ন কাজে ব্যবহৃত হচ্ছে, যা আমাদের everyday life কে আরও সহজ এবং উন্নত করছে।

Examples of Machine Learning Applications:

1. **Recommendation Systems:**
   Websites like **Netflix** বা **Amazon** আমাদের past activity দেখে নতুন movies বা products recommend করে।
2. **Image and Speech Recognition:**
   Facebook বা **Instagram** আমাদের ছবি বা videos থেকে automatically faces identify করতে পারে, এবং **Google Assistant** আমাদের voice commands বুঝতে পারে।
3. **Self-driving Cars:**
   Tesla এর মতো cars তাদের surroundings বুঝে সড়কে চলাচল করে, মানুষের সাহায্য ছাড়াই।
4. **Spam Email Detection:**
   Gmail আমাদের inbox থেকে spam emails automatically filter করে দেয়।
5. **Healthcare:**
   Doctors machine learning models ব্যবহার করে রোগের diagnosis করতে পারেন, যেমন cancer detection।

এইসব applications দ্বারা Machine Learning আমাদের জীবনে অনেক সুবিধা এনে দিয়েছে এবং বিভিন্ন industry তে এর ব্যবহার বাড়ছে।

3. Understanding supervised learning. Supervised Learning: target বলা থাকে ।

Supervised Learning হল এমন একটি মেশিন লার্নিং পদ্ধতি যেখানে **training data** তে **labeled instances** দেওয়া থাকে, এবং model (statistic formula) এই labeled data থেকে শিখে ভবিষ্যতে prediction বা classification করতে পারে। Colum হচ্ছে variable,

আপনার উদাহরণ অনুযায়ী, এখানে আমরা একটি **weather dataset** ব্যবহার করছি, যেখানে প্রতিটি তথ্যের সাথে একটি **labeled instance (Play)** দেয়া আছে, যা নির্দেশ করে যে সেদিন খেলাধুলা হবে কিনা।

Training Data:

Weather	Humidity	Wind	Play
Sunny	High	Weak	No
Sunny	Normal	Weak	No
Sunny	Normal	Strong	No
Cloudy	High	Weak	No
Cloudy	High	Strong	No
Cloudy	Normal	Strong	Yes
Cloudy	Normal	Weak	Yes
Rainy	High	Weak	Yes
Rainy	Normal	Strong	Yes
Rainy	Normal	Weak	Yes

Prediction:

এখন, আমরা একটি নতুন instance দেখতে পাচ্ছি:

Weather	Humidity	Wind	Play
Sunny	High	Strong	???

আমরা এখন model কে শিখানো training data থেকে **"Play"** এর label predict করতে চাই, যেহেতু এটা **supervised learning**।

Approach:

Model আমাদের দেওয়া data দেখে শিখেছে যে:

- যখন **Weather = Sunny**, **Humidity = High**, এবং **Wind = Weak**, তখন **Play = No**।
- যখন **Weather = Sunny**, **Humidity = Normal**, এবং **Wind = Weak** অথবা **Strong**, তখনও **Play = No**।

এখন, **Weather = Sunny**, **Humidity = High**, এবং **Wind = Strong** হওয়া সত্ত্বেও, আমাদের training data এর মধ্যে এরকম কোন সঠিক মিল পাওয়া যাচ্ছে না। তবে, আমাদের model সম্ভবত জানবে যে Sunny weather, High humidity এবং Strong wind সাধারণত খেলার জন্য উপযুক্ত নয়, কারণ অন্য সমস্ত Sunny instances এ Play ছিল No।

Prediction:

তাহলে, model এর prediction হতে পারে: **Play = No**।

উপসংহার:

এভাবে, supervised learning এর মাধ্যমে model training data থেকে শিখে **new instances** এর জন্য prediction করতে পারে।

4. Introduction to unsupervised learning.

unsupervised Learning: Target দেয়া থাকবে না ।

Unsupervised Learning হলো মেশিন লার্নিং এর একটি পদ্ধতি, যেখানে model কে কোন label বা class দেয়া হয় না, অর্থাৎ data তে কোন predefined output নেই। এই learning পদ্ধতিতে model **clustering** বা grouping করতে শেখে, যাতে সে সাদৃশ্যযুক্ত তথ্যের গ্রুপ তৈরি করতে পারে।

এটি সাধারণত **data grouping** বা **clustering** এর জন্য ব্যবহৃত হয়, যেমন data কে বিভিন্ন category তে ভাগ করা।

Example:

ধরা যাক, আমাদের কাছে কিছু **player statistics** রয়েছে, যার মধ্যে রয়েছে তাদের রান এবং উইকেটের সংখ্যা। আমাদের goal হলো **batsman**, **baller**, এবং **all-rounder** এর মধ্যে classification করা। যেহেতু এই data তে **label** দেওয়া হয়নি (এখানে জানা নেই কে batsman, কে baller এবং কে all-rounder), তাই আমরা **Unsupervised Learning** এর মাধ্যমে grouping বা clustering করব।

Data Table:

Runs	Wickets
10345	1
1768	2
8000	15
4567	12
5478	32
6876	2
1000	32
800	23
900	33
566	19
10872	0
7864	0
678	18
2345	2

Clustering:

- **Batsman:** সাধারণত যারা রান করেন কিন্তু উইকেটের সংখ্যা কম থাকে (অর্থাৎ উইকেটের সংখ্যা 1 বা 0 এর কাছাকাছি থাকে)।
- **Baller:** যারা উইকেট বেশি নেন কিন্তু রান কম করেন (অর্থাৎ উইকেট সংখ্যা বেশি, রান কম)।
- **All-rounder:** যারা রান এবং উইকেট দুইই ভালো করেন (অর্থাৎ, রান এবং উইকেট দুইটির মধ্যে একটি সামঞ্জস্য থাকে)।

এখন, **Unsupervised Learning** পদ্ধতিতে, model এই রান এবং উইকেটের ভিত্তিতে players কে বিভিন্ন গ্রুপে ভাগ করবে, যেমন:

- **Batsman:** কম উইকেট, বেশি রান।
  - উদাহরণ: **10345, 1768, 8000, 4567**
- **Ballers:** বেশি উইকেট, কম রান।
  - উদাহরণ: **5478, 6876, 1000, 800, 900**
- **All-rounders:** রান এবং উইকেট দুটোই ভালো।
  - উদাহরণ: **2345, 7864, 566, 678**

Conclusion:

Unsupervised Learning এর মাধ্যমে আমরা players কে **batsman**, **baller**, এবং **all-rounder** ক্যাটাগরিতে ভাগ করতে পারব, যেহেতু এখানে কোন label নেই এবং model রান এবং উইকেটের সম্পর্ক দেখে grouping বা clustering করবে।

এটি **K-Means Clustering** বা অন্যান্য clustering algorithms ব্যবহার করে করা যেতে পারে, যেখানে model বিভিন্ন data points কে cluster করবে তাদের similarities অনুযায়ী।

5. Overview of machine learning models.

Machine Learning Models হল অ্যালগোরিদম যা **data** থেকে শিখে এবং **prediction** বা **decision** নেয়। বিভিন্ন ধরনের মডেল রয়েছে, যা বিভিন্ন সমস্যা সমাধান করে।

1. Supervised Learning Models:

- **Definition:** Input-output pair এর সাথে model শিখে।
- **Examples:**
    - **Linear Regression** (continuous prediction)
    - **Logistic Regression** (classification)
    - **Decision Trees** (classification/regression)
    - **K-Nearest Neighbors (KNN)** (classification/regression)

2. Unsupervised Learning Models:

- **Definition:** Data এর hidden patterns খুঁজে বের করে, কোন label থাকে না।
- **Examples:**
    - **K-Means Clustering** (data clustering)
    - **PCA** (dimensionality reduction)
    - **Hierarchical Clustering** (clustering)

3. Reinforcement Learning Models: reinforcement learning: নিজে নিজে শিখবে ।

- **Definition:** Agent environment এর সাথে interact করে এবং feedback থেকে শিখে।
- **Examples:**
    - **Q-Learning**
    - **Deep Q-Networks (DQN)**
    - **Policy Gradient Methods**

Types Based on Problem Type:

- **Classification:** Discrete output (e.g., spam detection) – Examples: Logistic Regression, Decision Trees
- **Regression:** Continuous output (e.g., house price prediction) – Examples: Linear Regression, SVM
- **Clustering:** Grouping data (e.g., customer segmentation) – Examples: K-Means, DBSCAN
- **Anomaly Detection:** Identifying outliers – Examples: Isolation Forest, One-Class SVM

- **Dimensionality Reduction:** Reducing features – Examples: PCA, Autoencoders

Machine learning models data থেকে শিখে বিভিন্ন ধরনের সমস্যার সমাধান করে।

6. Data splitting: Training and test sets.

Training Data:

Training data হলো সেই data যা **model** শেখানোর জন্য ব্যবহৃত হয়। এখানে **input** এবং **expected output** থাকে, যাতে model শিখতে পারে কীভাবে সঠিক prediction করতে হয়।

Test Data:

Test data হলো সেই data যা model তৈরি হওয়ার পর **accuracy** পরীক্ষা করতে ব্যবহৃত হয়। এতে model এর সঠিকতা যাচাই করা হয়, যাতে নিশ্চিত হওয়া যায় যে model নতুন data দেখে সঠিক prediction করতে পারবে।

Splitting Data:

Data কে দুইটি ভাগে ভাগ করা হয়: **Training Set** এবং **Testing Set**। সাধারণত, 80% data training এর জন্য এবং 20% data testing এর জন্য ব্যবহার করা হয়।

7. Understanding K-fold cross-validation.

K-Fold Cross Validation একটি পদ্ধতি যেখানে data কে **K**টি সমান ভাগে ভাগ করা হয় এবং model এর **performance** পরীক্ষা করা হয়। প্রতিটি ভাগ একবার **testing data** হিসেবে এবং বাকি অংশগুলো **training data** হিসেবে ব্যবহৃত হয়। এটি **K** বার repeat করা হয়, যাতে model এর সঠিক performance পাওয়া যায় এবং overfitting কম হয়।

How K-Fold Cross Validation Works:

1. **Step 1:** Data কে **K**টি সমান ভাগে ভাগ করা হয় (যেমন, K = 5 হলে, data কে 5 ভাগে ভাগ করা হবে)।
2. **Step 2:** প্রথম ভাগকে **testing data** হিসেবে এবং বাকি 4টি ভাগকে **training data** হিসেবে ব্যবহার করা হয়। Model এর performance পরীক্ষা করা হয়।
3. **Step 3:** পরবর্তীতে দ্বিতীয় ভাগকে **testing data** হিসেবে ব্যবহার করা হয় এবং বাকি 4টি ভাগকে **training data** হিসেবে ব্যবহার করা হয়। আবার model পরীক্ষা করা হয়।
4. **Step 4:** এটি **K** বার repeat করা হয়, যেখানে প্রতিটি ভাগ একবার করে **testing data** হিসেবে ব্যবহৃত হয়।
5. **Step 5:** সবশেষে, প্রতিটি iteration এর **performance** গড়ে নিয়ে একটি সঠিক evaluation করা হয়।

Example:

ধরা যাক, আমাদের কাছে 10টি data points আছে এবং আমরা **K = 5** ব্যবহার করছি।

Data Points
1
2
3
4
5
6
7
8
9
10

Step-by-Step:

- **1st Fold:** Data 1, 2, 3, 4, 5, 6, 7, 8, 9 => Training, Data 10 => Testing
- **2nd Fold:** Data 1, 2, 3, 4, 5, 6, 7, 8, 10 => Training, Data 9 => Testing
- **3rd Fold:** Data 1, 2, 3, 4, 5, 6, 7, 9, 10 => Training, Data 8 => Testing
- **4th Fold:** Data 1, 2, 3, 4, 5, 6, 8, 9, 10 => Training, Data 7 => Testing
- **5th Fold:** Data 1, 2, 3, 4, 5, 7, 8, 9, 10 => Training, Data 6 => Testing

Model এর performance প্রতিটি fold এর পর গড়ে বের করা হয়, এবং শেষে গড়ে ফলাফল নেয়া হয়।

Summary:

K-Fold Cross Validation হলো একটি method যেখানে data কে **K** ভাগে ভাগ করে **K** বার model এর accuracy পরীক্ষা করা হয়, যাতে model এর performance আরো accurately জানা যায়।

K-Fold Cross-Validation এর মূল উদ্দেশ্য হলো model কে **সব ধরনের data** দেখে শিখতে সাহায্য করা।

ধরা যাক, ১০ জন ছেলের মধ্যে ৮ জন কালো এবং ২ জন ফর্সা। যদি **training data** তে ৮ জন কালো থাকে এবং **test data** তে ২ জন ফর্সা থাকে, তবে model ফর্সা ছেলেদের চিনতে পারবে না।

এজন্য **K-Fold Cross-Validation** ব্যবহার করা হয়, যেখানে data কে ভাগ করে প্রতিবার training ও testing করা হয়। এতে model সব ধরনের data (কালো, ফর্সা) দেখে শিখতে পারে এবং সঠিক prediction করতে সক্ষম হয়।

8. Handling underfitting and overfitting.

Underfitting:

Underfitting হয় যখন মডেল সঠিকভাবে শিখতে পারে না এবং **training data** থেকেও ভালো ফলাফল পায় না। এর মানে হল যে মডেল খুবই সাধারণ এবং **data** এর প্যাটার্ন বুঝতে পারছে না।

উদাহরণ:

ধরা যাক, আপনার কাছে একটি house price prediction model তৈরি করতে হবে এবং আপনি **linear regression** ব্যবহার করছেন। কিন্তু আপনার data যদি অনেক complex হয় (যেমন, house size, location, condition ইত্যাদি অনেক কিছু) এবং আপনি খুব সহজ একটি মডেল ব্যবহার করেন, তাহলে মডেল পুরো data এর complex pattern বুঝতে পারবে না এবং ফলস্বরূপ **underfitting** হবে।

Symptoms of Underfitting:

- Training data তে poor accuracy
- Model overly simple (like a straight line in a complex scenario)

Overfitting:

Overfitting হয় যখন মডেল **training data** খুব ভালোভাবে শিখে, কিন্তু **new data** বা **test data** তে সঠিকভাবে কাজ করতে পারে না। মডেল training data তে সব noise এবং অপ্রয়োজনীয় details শিখে ফেলে, যা ভবিষ্যতে নতুন data এর জন্য সমস্যার সৃষ্টি করে।

উদাহরণ:

ধরা যাক, আপনার কাছে ১০০টা house price এর data আছে এবং আপনি decision tree ব্যবহার করছেন। যদি tree খুব গভীর হয়ে যায় এবং training data তে perfectly fit হয়ে যায়, তবে model সম্ভবত training data এর কিছু অপ্রয়োজনীয় pattern ও শিখে ফেলবে, যেগুলো **test data** তে কাজ করবে না। ফলে **overfitting** হবে।

Symptoms of Overfitting:

- Training data তে খুব ভালো accuracy (100%)
- Test data তে poor performance

Handling Underfitting and Overfitting:

- Underfitting:
  - মডেলকে **complex** করতে হবে, যেমন **more features**, **better algorithms**, বা **more training data**।
  - মডেলকে **tune** করা।
- Overfitting:
  - **Simplify** the model, যেমন decision tree এর গভীরতা কমানো।
  - **Regularization** techniques ব্যবহার করা (যেমন L1, L2 regularization)।
  - **Cross-validation** ব্যবহার করা।
  - বেশি **training data** ব্যবহার করা।

সারাংশ:

- **Underfitting** তখন হয় যখন মডেল যথেষ্ট শিখতে পারে না এবং তার performance খারাপ থাকে।
- **Overfitting** তখন হয় যখন মডেল training data তে খুব ভালো শিখে, কিন্তু নতুন data তে খারাপ পারফর্ম করে।
- **Solution:** Underfitting এর জন্য মডেলকে complex করা, আর overfitting এর জন্য মডেলকে simple ও regularize করা।

9. Confusion matrix metrics: Precision, recall, and F1 score.

Confusion Matrix:

Confusion Matrix হল একটি টেবিল যা মডেল এর পারফরম্যান্স পরিমাপ করতে সাহায্য করে, বিশেষত ক্লাসিফিকেশন মডেলগুলোর ক্ষেত্রে। এটি actual এবং predicted ফলাফলগুলোর মধ্যে তুলনা করতে ব্যবহৃত হয়। কনফিউশন ম্যাট্রিক্সে মোট ৪টি উপাদান থাকে, যা মডেল এর সঠিকতা ও ভুল prediction গুলি বের করতে সাহায্য করে।

Confusion Matrix এর উপাদান:

1. **True Positive (TP):** মডেল সঠিকভাবে পজিটিভ (positive) ক্লাস সনাক্ত করেছে।
2. **True Negative (TN):** মডেল সঠিকভাবে নেগেটিভ (negative) ক্লাস সনাক্ত করেছে।
3. **False Positive (FP):** মডেল ভুলভাবে পজিটিভ (positive) ক্লাস সনাক্ত করেছে, অথচ এটি নেগেটিভ (negative) ছিল।
4. **False Negative (FN):** মডেল ভুলভাবে নেগেটিভ (negative) ক্লাস সনাক্ত করেছে, অথচ এটি পজিটিভ (positive) ছিল।

Confusion Matrix Structure:

	Predicted Positive (P)	**Predicted Negative (N)**
Actual Positive (P)	True Positive (TP)	False Negative (FN)
Actual Negative (N)	False Positive (FP)	True Negative (TN)

Confusion Matrix টেবিল:

	Predicted Positive (P)	**Predicted Negative (N)**
Actual Positive (P)	TP = 45	FN = 5
Actual Negative (N)	FP = 5	TN = 45

Confusion Matrix থেকে মেট্রিক্স বের করা যায়:

1. **Accuracy (সঠিকতা):**

   Accuracy মডেল এর মোট সঠিক ভবিষ্যদ্বাণী এবং ভুল ভবিষ্যদ্বাণীর অনুপাত।

   Formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

   Calculation:

$$Accuracy = \frac{45 + 45}{45 + 45 + 5 + 5} = \frac{90}{100} = 0.9$$

2. **Precision (প্রিসিশন):**

Precision মাপে, মোট **predicted positive** এর মধ্যে কতটুকু আসলে **positive** ছিল।

Formula:

$$Precision = \frac{TP}{TP + FP}$$

Calculation:

$$Precision = \frac{45}{45 + 5} = \frac{45}{50} = 0.9$$

3. **Recall (রিকল) বা Sensitivity:**

Recall মাপে, আসলে **positive** কতটুকু সঠিকভাবে **predict** হয়েছে।

Formula:

$$Recall = \frac{TP}{TP + FN}$$

Calculation:

$$Recall = \frac{45}{45 + 5} = \frac{45}{50} = 0.9$$

4. **F1-Score:**

F1-Score হলো Precision এবং Recall এর মধ্যে একটি ভারসাম্য।

Formula:

$$F1\,Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Calculation:

$$F1\,Score = 2 \times \frac{0.9 \times 0.9}{0.9 + 0.9} = 2 \times \frac{0.81}{1.8} = 0.9$$

5. **False Positive Rate (FPR):**

FPR মাপে, মডেল কতটা ভুলভাবে নেগেটিভ ক্লাসকে পজিটিভ হিসেবে চিহ্নিত করেছে।

Formula:

$$FPR = \frac{FP}{FP + TN}$$

Calculation:

$$FPR = \frac{5}{5 + 45} = \frac{5}{50} = 0.1$$

6. **False Negative Rate (FNR):**

FNR মাপে, মডেল কতটা ভুলভাবে পজিটিভ ক্লাসকে নেগেটিভ হিসেবে চিহ্নিত করেছে।

Formula:

$$FNR = \frac{FN}{FN + TP}$$

Calculation:

$$FNR = \frac{5}{5+45} = \frac{5}{50} = 0.1$$

Summary Table:

Metric	Formula	What it Measures	Example Calculation
Precision	$\frac{TP}{TP+FP}$	How many predicted positives are actual positives	$\frac{45}{45+5} = 0.9$
Recall	$\frac{TP}{TP+FN}$	How many actual positives are correctly predicted	$\frac{45}{45+5} = 0.9$
F1 Score	$2 \times \frac{Precision \times Recall}{Precision + Recall}$	Balance between Precision and Recall	$2 \times \frac{0.9 \times 0.9}{0.9+0.9} = 0.9$

Conclusion:

- **Precision** tells you how accurate the positive predictions are.
- **Recall** tells you how well you identified all the actual positive cases.
- **F1 Score** provides a balance between Precision and Recall, especially when you need to consider both equally.

These metrics help in evaluating a model's performance, especially in **imbalanced data sets** where accuracy alone might not give the full picture.

~ CODE ~

এই ডেটাসেটের মধ্যে তিনটি কলাম রয়েছে:

1. **Mileage**: গাড়ির কিলোমিটার (মাইলেজ) যে কত দূর চালানো হয়েছে, অর্থাৎ গাড়ির ব্যবহৃত দূরত্ব।
2. **Age (yrs)**: গাড়ির বয়স (বছর)।
3. **Sell Price ($)**: গাড়ির বিক্রির মূল্য (ডলার).

আপনি যদি এই ডেটা ব্যবহার করে মডেল তৈরির কাজ করতে চান (যেমন: গাড়ির বিক্রির মূল্য ভবিষ্যদ্বাণী করা), তাহলে আপনি **Mileage** এবং **Age** কে ফিচার হিসেবে ব্যবহার করবেন এবং **Sell Price** হবে লক্ষ্য বা আউটপুট (target variable)।

উদাহরণঃ

Features:

- Mileage (গাড়ির কিলোমিটার)
- Age (গাড়ির বয়স)

Target Variable:

- Sell Price (বিক্রির মূল্য)

উদাহরণ হিসেবে, যদি আপনি Python ব্যবহার করে মডেল তৈরি করতে চান:

1. **ডেটা লোড এবং প্রস্তুতি:** প্রথমে এই ডেটাটি পাণ্ডাস ডেটাফ্রেমে লোড করতে হবে।

```
import pandas as pd
# ডেটা তৈরি করা
data = {
    "Mileage": [69000, 35000, 57000, 22500, 46000, 59000, 52000, 72000, 91000, 67000, 83000, 79000, 59000, 58780, 82450, 25400, 28000, 69000, 87600, 52000],
    "Age(yrs)": [6, 3, 5, 2, 4, 5, 5, 6, 8, 6, 7, 7, 5, 4, 7, 3, 2, 5, 8, 5],
    "Sell Price($)": [18000, 34000, 26100, 40000, 31500, 26750, 32000, 19300, 12000, 22000, 18700, 19500, 26000, 27500, 19400, 35000, 35500, 19700, 12800, 28200]
}
# ডেটা ফ্রেম তৈরি
df = pd.DataFrame(data)
```

2. **ফিচার এবং টার্গেট বিভক্ত করা:**

```
# ফিচার এবং টার্গেট ভ্যারিয়েবল তৈরি
X = df[["Mileage", "Age(yrs)"]]  # ফিচার
y = df["Sell Price($)"]          # টার্গেট
```

3. **মডেল ট্রেনিং এবং পরীক্ষণের জন্য ডেটা ভাগ করা:**

```
from sklearn.model_selection import train_test_split
# ডেটা ভাগ করা
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# ট্রেনিং ডেটা এবং টেস্ট ডেটার আউটপুট
print(X_train)
print(X_test)
print(y_train)
```

```
print(y_test)
```

4. **মডেল প্রশিক্ষণ করা (যেমন: লিনিয়ার রিগ্রেশন):**

```
from sklearn.linear_model import LinearRegression


# মডেল তৈরি করা
model = LinearRegression()
# মডেল প্রশিক্ষণ
model.fit(X_train, y_train)
# পূর্বাভাস করা
y_pred = model.predict(X_test)


# পূর্বাভাস ফলাফল
print(y_pred)
```

সারাংশ:

- আপনি **Mileage** এবং **Age** ফিচার গুলো ব্যবহার করে **Sell Price** পূর্বাভাস করার জন্য একটি মডেল তৈরি করতে পারেন।
- **train_test_split()** ফাংশন দিয়ে ডেটা ভাগ করবেন, এর মাধ্যমে ৮০% ডেটা ট্রেনিংয়ের জন্য এবং ২০% ডেটা পরীক্ষার জন্য ব্যবহার হবে।

0, 42, 123 random_state হচ্ছে parameter. ran random_state এর default মান হচ্ছে 42.