Cricket T20 Dataset

Using train_test_split where random state=42

Import libraries

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

Loading Dataset

```
# Load the Boston Housing dataset
df = pd.read_csv('/content/Cricket T20.csv')
df.head(5)
```

	Mat	Inns	NO	Runs	HS	Ave	BF	SR	100	50	0	4s	6s
0	75	70	20	2633	94	52.66	1907	138.07	0	24	2	247	71
1	104	96	14	2633	118	32.10	1905	138.21	4	19	6	234	120
2	83	80	7	2436	105	33.36	1810	134.58	2	15	2	215	113
3	111	104	30	2263	75	30.58	1824	124.06	0	7	1	186	61
4	71	70	10	2140	123	35.66	1571	136.21	2	13	3	199	91

Next steps: Generate code with df View recommended plots

```
# Select all columns from the dataframe except the last one and assign them to variable 'X'
X = df.iloc[:, 0:-1]
# Select the last column from the dataframe and assign it to variable 'y'
y = df.iloc[:, -1]
```

Creating a linear regression model

```
# Create a linear regression model
model = LinearRegression()
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=46)
model.fit(x_train,y_train)
y_preds=model.predict(x_test)
print("Average MSE:", mean_squared_error(y_test,y_preds))
```

```
Average MSE: 41.27475402467207
```

Using train_test_split where random state=2

```
X = df.iloc[:,0:-1]
y = df.iloc[:,-1]
```

Creating a linear regression model

```
# Create a linear regression model
model = LinearRegression()
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=3)
model.fit(x_train,y_train)
y_preds=model.predict(x_test)
print("Average MSE:", mean_squared_error(y_test,y_preds))
```

```
Average MSE: 55.920168666648806
```

Leave one-out cross validation

import libraries

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import LeaveOneOut, cross_val_score


X = df.iloc[:,0:-1]
y = df.iloc[:,-1]
```

Creating a linear regression model and LeaveOneOut cross validator

```
# Create a linear regression model
model = LinearRegression()
# Create a LeaveOneOut cross-validator
loo = LeaveOneOut()
# Use cross_val_score for the dataset with the model and LOOCV
# This will return the scores for each iteration of LOOCV
scores = cross_val_score(model, X, y, cv=loo, scoring='neg_mean_squared_error')
mse_scores = -scores # Invert the sign of the scores
# Print the mean MSE over all LOOCV iterations
print("Mean MSE:", mse_scores.mean())
```

```
Mean MSE: 69.50601909467493
```

Implementing K-fold cross validation

Import Libraries

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import KFold
import pandas as pd


X = df.iloc[:,0:-1]
y = df.iloc[:,-1]
```

Creating a linear regression model and Initialize the KFold parameters

```
# Initialize a Linear Regression model
model = LinearRegression()
# Initialize the KFold parameters
kfold = KFold(n_splits=10, shuffle=True, random_state=2)
# Use cross_val_score on the model and dataset
scores = cross_val_score(model, X, y, cv=kfold, scoring='r2')
print("R2 scores for each fold:", scores)
print("Mean R2 score across all folds:", scores.mean())
```

```
R2 scores for each fold: [0.91546422 0.61088173 0.96031469 0.94039158 0.7886418  0.79880163
 0.64326403 0.95343645 0.80674918 0.81862228]
Mean R2 score across all folds: 0.823656758561729
```

Implementing Stratified K-fold cross validation

```
from sklearn.datasets import load_iris
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.linear_model import LogisticRegression
```

Creating a Logistic Regression model and StratifiedKFold object

```
# Load iris dataset
data = load_iris()
X, y = data.data, data.target
# Create a Logistic Regression model
model = LogisticRegression(max_iter=10000, random_state=42)
# Create StratifiedKFold object
skf = StratifiedKFold(n_splits=5, random_state=42, shuffle=True)
# Perform stratified cross validation
scores = cross_val_score(model, X, y, cv=skf, scoring='accuracy')
# Print the accuracy for each fold

print("Accuracies for each fold: ", scores)
print("Mean accuracy across all folds: ", scores.mean())
```

```
Accuracies for each fold:  [1.         0.96666667 0.93333333 1.         0.93333333]
Mean accuracy across all folds:  0.9666666666666668
```