Data Science B

Machine Learning Model Deployment

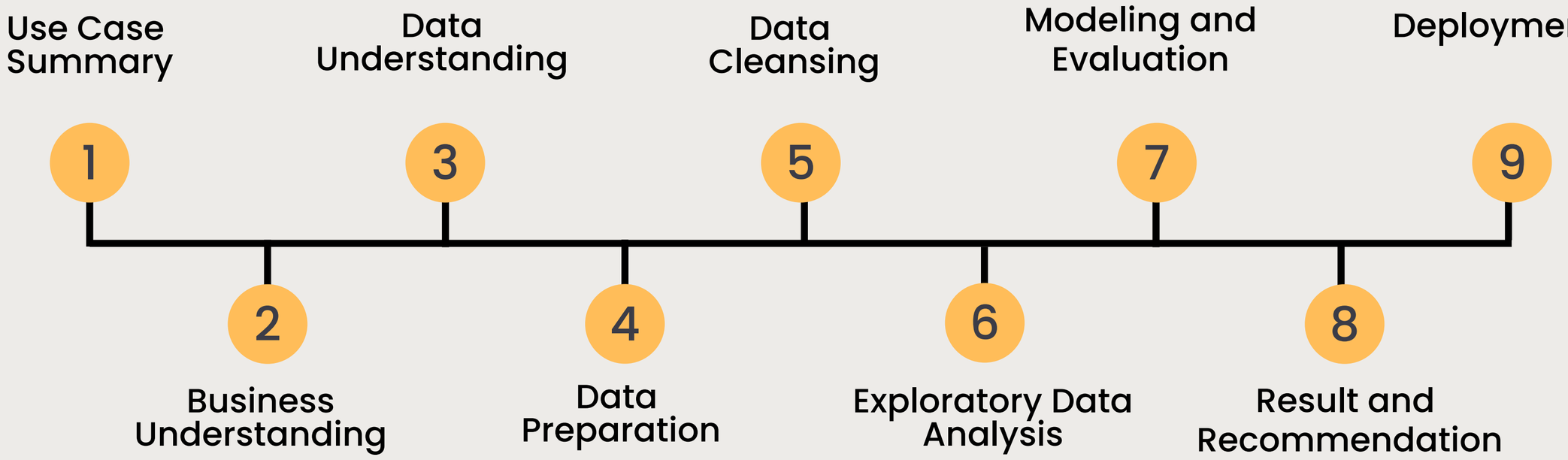
Model Tracking Using MLflow

By Group 3

Hello, This is Group 3!

- Agung Prayogi
- Axel Eldrian H.
- Dian Maulida
- Dwi Fitriawati F.
- Yesaya Arya D. K.

Workflow


Use Case Summary
1
Business Understanding
2
Data Understanding
3
Data Preparation
4
Data Cleansing
5
Exploratory Data Analysis
6
Modeling and Evaluation
7
Result and Recommendation
8
Deployment
9

Use Case Summary

Objective Statements

- Get an insight into how much the company spends on the marketing budget
- Get an insight into how much sales based on the marketing budget
- Get an understanding about the correlation between marketing budget and sales
- Get an understanding about marketing strategy or planning based on the model prediction
- Create models to predict sales with simple linear regression
- Deploy the model using MLflow deployment

Challenges

- There is no explanation for each column

Methodology / Analytic Technique

- Descriptive analysis
- Graph analysis
- Modeling using Simple Linear Regression
- Deployment using MLflow

Business Benefit

- Help Business to make decision about marketing strategy based on the prediction

Expected Outcome

- Get to know about how much the company spend on marketing budget
- Get to know about how much sales based on marketing budget
- Get to know about the correlation between marketing budget and sales
- Recommendation based on model prediction
- Making models to predict sales
- Deploy the model using MLflow

Business Understanding

Business Understanding

A marketing budget is the sum of money a company assigns to marketing projects (paid advertising, sponsored content, etc.) aimed at product promotion. It helps startups and established companies manage resources efficiently and achieve business goals.

This case has some business questions using the data:

- How much does the company spends on marketing budget?
- How much sales are based on the marketing budget?
- Is there any correlation between marketing budget and sales?
- How about the recommendation based on the model prediction?

Data Understanding

Source Data

- Source Data:
- https://www.kaggle.com/code/devzohaib/simple-linear-regression/notebook
- 200 rows
- 2 columns

Data Dictionary

- Tv = marketing budget
- Sales = total sales

Data Preparation

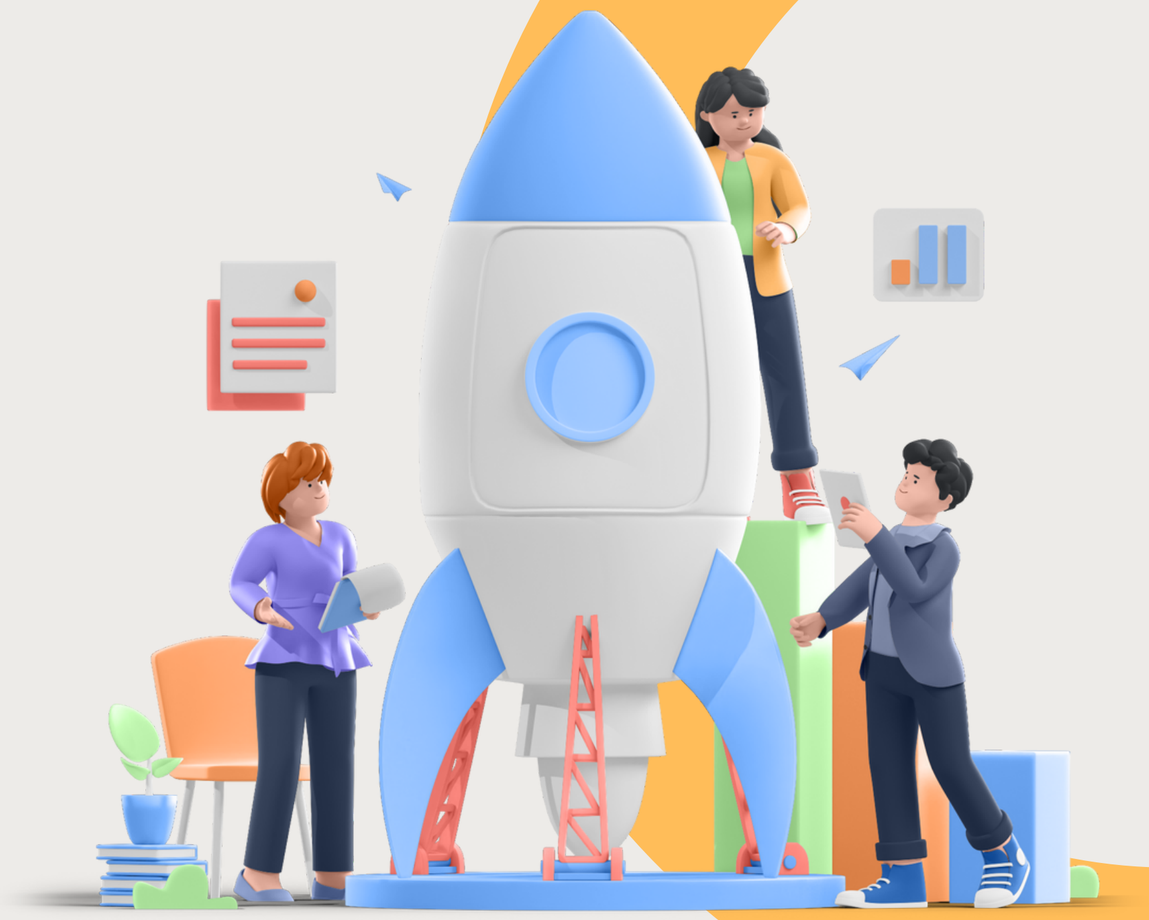
Packages Version:

- Pandas, numpy, matplotlib, seaborn, sklearn, and mlflow

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

import warnings
warnings.filterwarnings('ignore')
```

```
import logging
import mlflow.sklearn
import mlflow
from urllib.parse import urlparse
```



Data Cleansing

Data is clean

The data type is all correct and there is no missing value.

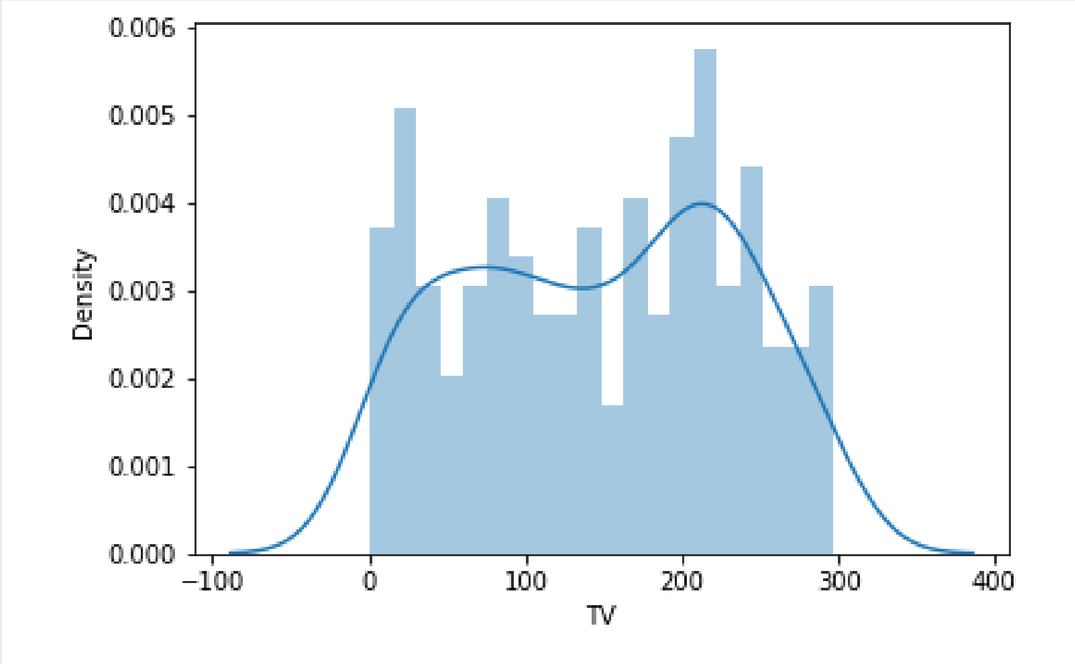
Exploratory Data Analysis

Statistic Numerical Data

	TV	Sales
count	200.000000	200.000000
mean	147.042500	14.022500
std	85.854236	5.217457
min	0.700000	1.600000
25%	74.375000	10.375000
50%	149.750000	12.900000
75%	218.825000	17.400000
max	296.400000	27.000000

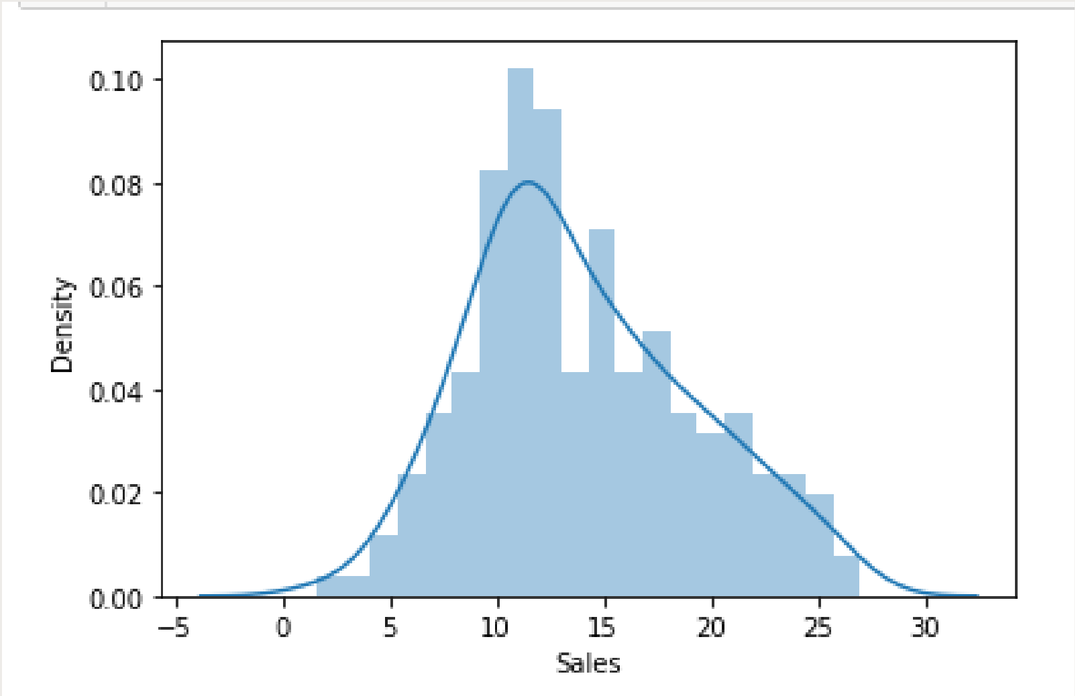
From the analysis of *tvmarketing.csv* data which contains 200 lines, the average cost spent on the TV marketing budget is **$147.04** with the lowest cost spending **$0.7** and the highest cost of **$296.4**. Also, the average sales are **$14.02** with the lowest being **$1.6** and the highest is **$27**.

Distribution of Marketing Budget



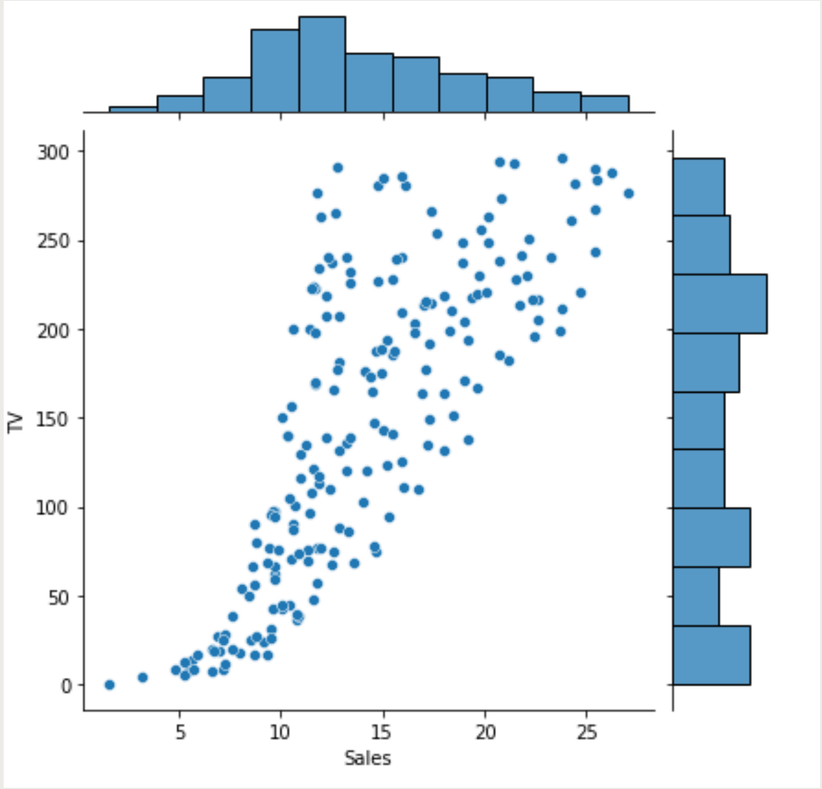
Based on the graph above distribution of marketing budget 'TV' has the highest density of marketing Budget which is $200 dollars

Distribution of Sales



Based on the data above, we can see that 'Sales' are mostly or have the highest density at values of $10-15

The Correlation between TV and Sales



Based on the graph, we can see that the Sales and Marketing Budget of TV has a correlation, where the higher the marketing costs of the TV, the greater the selling price of the TV. However, from the graph above there is also data that is known to have a low selling price but high marketing costs.

Feature Engineering

Scaling the data

MinMax Scaler

We duplicate dataset to do scaling using MinMaxScaler

Features in the data have different value ranges. By normalizing the data using MinMaxScaler, the data will have the same scale and the model can learn faster and the accuracy of the model can increase.

```
# MinMaxScaler
df_copy = df.copy()
scaler = MinMaxScaler()
df_copy[['TV']] = scaler.fit_transform(df_copy[['TV']])
df
```

	TV	Sales
0	230.1	22.1
1	44.5	10.4
2	17.2	9.3
3	151.5	18.5
4	180.8	12.9
...	...	...
195	38.2	7.6
196	94.2	9.7
197	177.0	12.8
198	283.6	25.5
199	232.1	13.4

200 rows × 2 columns

Modeling

Without Hyperparameter Tuning

Preprocessing Modeling

Split and train the data

```
# split and train dataset first, we will predict the Sales data
x = df.drop(['Sales'],axis=1)
y = df['Sales']

# used 0 for train size because data because the amount of data in the dataset is quite a lot
x_train, x_test, y_train, y_test =  train_test_split(x, y, train_size = 0.8, random_state = 42)
```

✓ 0.7s

We want to predict, data Sales from dataset. and we used 0.8 for **train_size** because the amount of data in tvmarketing.csv is quiet a lot.

Simple Linear Regression

```
regressor = LinearRegression() # Regression with default parameter
regressor.fit(x_train, y_train) # fit data train to Linear Regression
regressor.coef_
y_pred = regressor.predict(x_test) # get data predict
```

✓ 0.9s

After get predict data for **Sales** column, compare them with Actual data value

```
result = pd.DataFrame({'Actual' : y_test, 'Predict': y_pred})
result.head()
```

✓ 0.1s

	Actual	Predict
95	16.9	14.717944
15	22.4	16.211548
30	21.4	20.748197
158	7.3	7.664036
128	24.7	17.370139

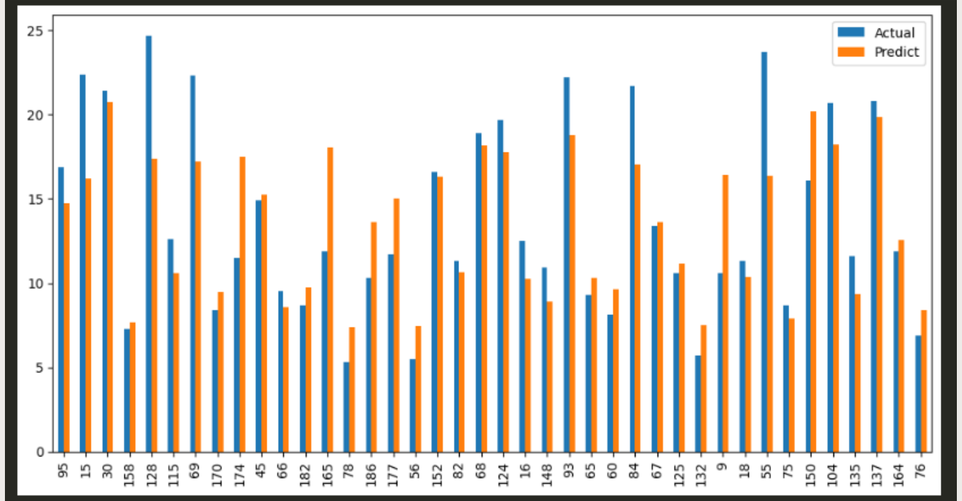
for the first case, we do Linear Regression without *hypertuning parameter*, so we use parameter as default. after fit dataset with regression and get the predict data from **Sales** column, compare them with **actual** data value.

Compare with Bar Chart and Scatter Plot, to see a pattern of relationship between 2 variable

Also we wan see the difference using **bar chart**

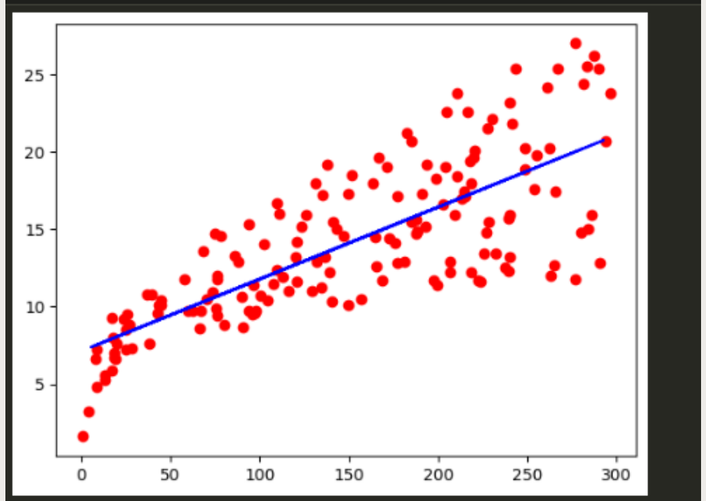
```
sult.plot(kind = 'bar', figsize = (12,6))
plt.show()
```

✓ 0.4s



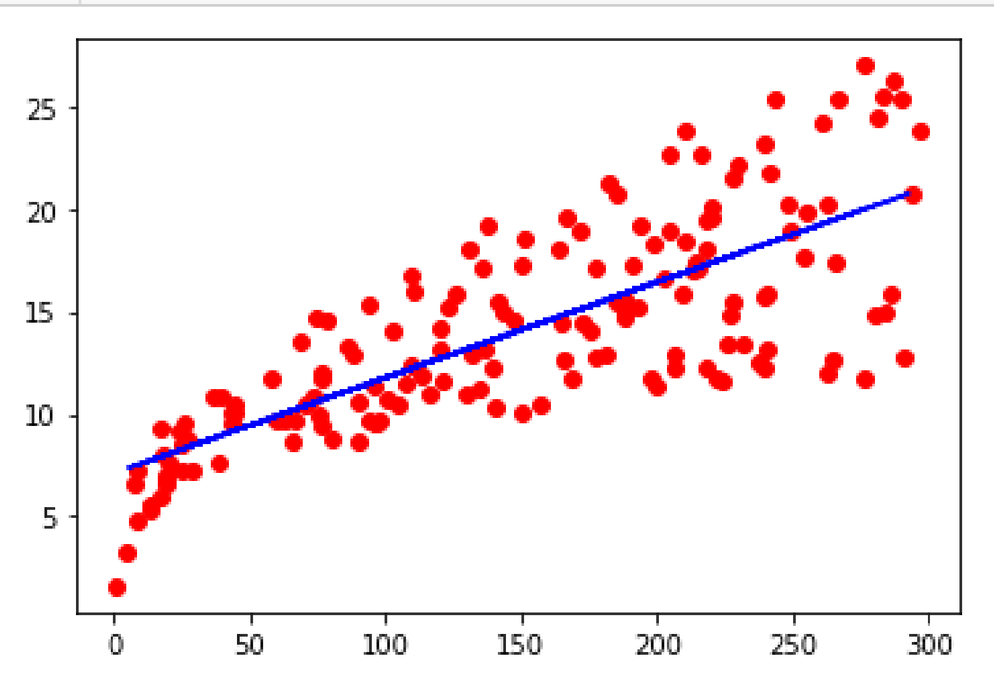

```
plt.scatter(x_train, y_train, color='red')
plt.plot(x_test.values, y_pred, color='blue')
plt.show()
```

✓ 0.2s



Compare them using **bar chart** with mathplotlib or **Scatter** plot to get detail.

Analysis



From the graph, it is known that there is a positive relationship between the train data and the predictive data, the red dot indicates the train data which is close to the blue line (predictive data). This shows that the model used is good enough for sales predictions from the tvmarketing dataset.

Evaluate Model

```
MAE = mean_absolute_error(y_test, y_pred)
MAE
```

```
2.444420003751042
```

```
MAPE = mean_absolute_percentage_error(y_test, y_pred)
MAPE
```

```
0.1866874631359211
```

```
RMSE = np.sqrt(mean_squared_error(y_test, y_pred))
RMSE
```

```
3.194472431998898
```

```
R2 = r2_score(y_test, y_pred)
R2
```

```
0.6766954295627077
```

Based on the results of the evaluation that has been carried out, we can see that the MAE value is 2.44, the smaller the MAE value, the more accurate the model used. The MAPE value is 0.18 or 18%, this figure indicates the percentage error of the predictions made, the smaller the percentage error value in MAPE, the more accurate the forecasting results, for MAPE is 18% it is known that the predictions made are still classified as good for prediction. The RMSE value is 3.19, RMSE indicates that the variation in values produced by a forecast model is close to the variation in the observed value. RMSE measures how different the sets of values are. The smaller the RMSE value, the closer the predicted and detected values are.

The value of R2 is 0.67 or 67%, R square has a value between 0 – 1 provided that the closer to number one means the better, R2 has a value of 0.67, meaning that 67% of the distribution of the dependent variable can be explained by the independent variable. The remaining 33% cannot be explained by independent variables or can be explained by variables outside the independent variables (component errors).

Modeling

With Hyperparameter Tuning

Preprocessing Modeling

Split and train the data

```
x_new = df_copy.drop(['Sales'],axis=1)
y_new = df_copy['Sales']
```

```
x_train_new, x_test_new, y_train_new, y_test_new =  train_test_split(x_new, y_new, train_size = 0.8, random_state = 42)
```

We want to predict, data Sales from dataset. and we used 0.8 for **train_size** because the amount of data in tvmarketing.csv is quiet a lot

Hyperparameter tuning

```
scores = cross_val_score(regressor, x_train_new, y_train_new, scoring='r2', cv=3)
scores
```

```
array([0.67562287, 0.49242015, 0.56765656])
```

```
regressor.get_params()
```

```
{'copy_X': True,
 'fit_intercept': True,
 'n_jobs': None,
 'normalize': 'deprecated',
 'positive': False}
```

```
parameters = {
    'copy_X' : [True, False],
    'fit_intercept' : [True, False],
    'n_jobs' : [None, -1],
    'normalize' : [True, False],
    'positive' : [True, False]
}
```

```
regressor_new = LinearRegression(positive=True, normalize=True, n_jobs=None, fit_intercept=True, copy_X=True)
```

```
model_new = regressor_new.fit(x_train_new, y_train_new)
```

```
y_pred_new = regressor_new.predict(x_test_new)
```

For the first case, we do Linear Regression with *hypertuning parameter*, so we use get the best parameter first, then fit to the modelt. after fit dataset with regression and get the predict data from **Sales** column, compare them with **actual** data value

Simple Linear Regression

```
regressor_new = LinearRegression(positive=True, normalize=True, n_jobs=None, fit_intercept=True, copy_X=True)
```

```
model_new = regressor_new.fit(x_train_new, y_train_new)
```

```
y_pred_new = regressor_new.predict(x_test_new)
```

After fit with the best parameters, dataset with regression and get the predict data from **Sales** column, compare them with **actual** data value

Evaluate Model

```
MAE_new = mean_absolute_error(y_test_new, y_pred_new)
MAE_new
```

```
2.4444200037510426
```

```
MAPE_new = mean_absolute_percentage_error(y_test_new, y_pred_new)
MAPE_new
```

```
0.18668746313592113
```

```
RMSE_new = np.sqrt(mean_squared_error(y_test_new, y_pred_new))
RMSE
```

```
3.194472431998898
```

```
R2_new = r2_score(y_test_new, y_pred_new)
R2_new
```

```
0.6766954295627075
```

Based on the results of the evaluation that has been carried out, we can see that the MAE value is 2.44, the smaller the MAE value, the more accurate the model used. The MAPE value is 0.18 or 18%, this figure indicates the percentage error of the predictions made, the smaller the percentage error value in MAPE, the more accurate the forecasting results, for MAPE is 18% it is known that the predictions made are still classified as good for prediction. The RMSE value is 3.19, RMSE indicates that the variation in values produced by a forecast model is close to the variation in the observed value.

RMSE measures how different the sets of values are. The smaller the RMSE value, the closer the predicted and detected values are. The value of R2 is 0.67 or 67%, R square has a value between 0 – 1 provided that the closer to number one means the better, R2 has a value of 0.67, meaning that 67% of the distribution of the dependent variable can be explained by the independent variable. The remaining 33% cannot be explained by independent variables or can be explained by variables outside the independent variables (component errors).

Comparison the Evaluation Model

There is no difference so whether or not using hyperparameters is good.

The model that was performed by hyperparameter tuning showed no difference in values for MAE, MAPE, and RSME, which means that the use of hyperparameters in this dataset does not have to be done.

	MAE	Values
0	Without	2.44442
1	with	2.44442

	RMSE	Values
0	Without	3.194472
1	with	3.194472

	MAPE	Values
0	Without	0.186687
1	with	0.186687

	R2	Values
0	Without	0.676695
1	with	0.676695

Result

- From the analysis of tvmarketing data which contains 200 lines, the average cost spent on the TV marketing budget is $147.04 with the lowest cost spending $0.7 and the highest cost $296.4, and the average sales is$14.02 with a lowest is $1.6 and the highest is $27.
- Based on the graph above distribution of marketing budget 'TV' has the highest density of marketing Budget which is $200 dollars
- Based on the data above, we can see that 'Sales' are mostly or have the highest density at values of $10-15
- Based on the graph above, we can see that the Sales and Marketing Budget of TV has a correlation, where the higher the marketing costs , the greater the sales. However, from the graph above there is also data that has high marketing costs but has low sales
- From the graph above we can see that there is a positive relationship between the train data and the predictive data, the red dot indicates the train data which is close to the blue line (predictive data). This shows that the model used is good enough for sales predictions from the tvmarketing dataset.

- Based on the results of the evaluation that has been carried out, we can see that the MAE value is 2.44, the smaller the MAE value, the more accurate the model used. The MAPE value is 0.18 or 18%, this figure indicates the percentage error of the predictions made, the smaller the percentage error value in MAPE, the more accurate the forecasting results, for MAPE is 18% it is known that the predictions made are still classified as good for prediction. The RMSE value is 3.19, RMSE indicates that the variation in values produced by a forecast model is close to the variation in the observed value. RMSE measures how different the sets of values are. The smaller the RMSE value, the closer the predicted and detected values are. The value of R2 is 0.67 or 67%, R square has a value between 0 – 1 provided that the closer to number one means the better, R2 has a value of 0.67, meaning that 67% of the distribution of the dependent variable can be explained by the independent variable. The remaining 33% cannot be explained by independent variables or can be explained by variables outside the independent variables (component errors).
- There is no difference in the evaluation value of the data that we do the hyperparameter tuning and not
- The model that was performed by hyperparameter tuning showed no difference in values for MAE, MAPE, and RSME, which means that the use of hyperparameters in this dataset does not have to be done.

Recommendation

Companies can **optimize marketing budget costs for marketing in order to get high profits**, optimizing marketing costs does not have to use large costs because it can be seen from the data that there is a large marketing budget but low selling prices

Deployment

MLflow Code

```
# Data Profiling

# Import Packages

import logging
import mlflow.sklearn
import mlflow
from urllib.parse import urlparse
import os
import warnings
import sys

import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_percentage_error, r2_score

from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error

import warnings
warnings.filterwarnings('ignore')
```

MLflow Code

```
logging.basicConfig(level=logging.WARN)
logger = logging.getLogger(__name__)

if __name__ == "__main__":
    warnings.filterwarnings("ignore")
    np.random.seed(40)

# Load Dataset

df = pd.read_csv('tvmarketing.csv')

# Preprocessing Modeling

x = df.drop(['Sales'], axis=1)
y = df['Sales']

# Splitting and Train Data

X_train, X_test, y_train, y_test = train_test_split(
    x, y, train_size=0.8, random_state=42)
```

MLflow Code

```python
# Eval Metrics

def eval_metrics(actual, pred):
    rmse = np.sqrt(mean_squared_error(actual, pred))
    mae = mean_absolute_error(actual, pred)
    mape = mean_absolute_percentage_error(actual, pred)
    r2 = r2_score(actual, pred)
    return rmse, mae, mape, r2

# Modeling

with mlflow.start_run():
    lr = LinearRegression()
    lr.fit(X_train, y_train)

    y_pred = lr.predict(X_test)

    (rmse, mae, mape, r2) = eval_metrics(y_test, y_pred)

    print("  RMSE: %s" % rmse)
    print("  MAE: %s" % mae)
    print("  MAPE: %s" % mape)
    print("  R2: %s" % r2)
```

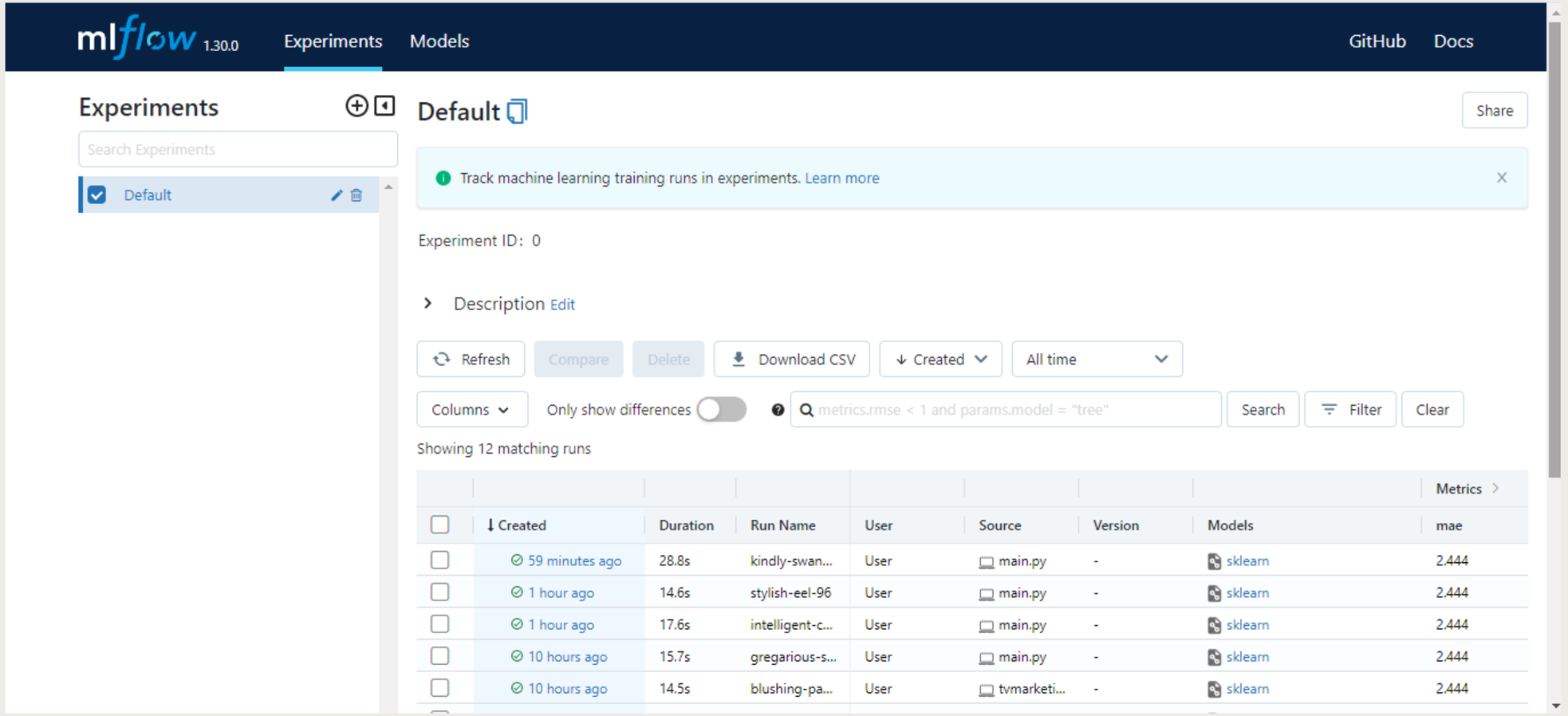
MLflow Code

```
    mlflow.log_metric("rmse", rmse)
    mlflow.log_metric("mae", mae)
    mlflow.log_metric("mape", mape)
    mlflow.log_metric("r2", r2)

    tracking_url_type_store = urlparse(mlflow.get_tracking_uri()).scheme

    if tracking_url_type_store != "file":
        mlflow.sklearn.log_model(
            lr, "model", registered_model_name="Linear Regression")
    else:
        mlflow.sklearn.log_model(lr, "model")
```

MLflow UI


mlflow 1.30.0
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
Default
Default
Share
Track machine learning training runs in experiments. Learn more
Experiment ID: 0
Description Edit
Refresh
Compare
Delete
Download CSV
Created
All time
Columns
Only show differences
metrics.rmse < 1 and params.model = "tree"
Search
Filter
Clear
Showing 12 matching runs
Metrics
Created
Duration
Run Name
User
Source
Version
Models
mae
59 minutes ago 28.8s kindly-swan... User main.py - sklearn 2.444
1 hour ago 14.6s stylish-eel-96 User main.py - sklearn 2.444
1 hour ago 17.6s intelligent-c... User main.py - sklearn 2.444
10 hours ago 15.7s gregarious-s... User main.py - sklearn 2.444
10 hours ago 14.5s blushing-pa... User tvmarketi... - sklearn 2.444

MLflow is a platform to streamline machine learning development, including tracking experiments, packaging code into reproducible runs, and sharing and deploying models. MLflow offers a set of lightweight APIs that can be used with any existing machine learning application or library (TensorFlow, PyTorch, XGBoost, etc), wherever you currently run ML code (e.g. in notebooks, standalone applications or the cloud). MLflow's current components are:

- MLflow Tracking: An API to log parameters, code, and results in machine learning experiments and compare them using an interactive UI.
- MLflow Projects: A code packaging format for reproducible runs using Conda and Docker, so you can share your ML code with others.
- MLflow Models: A model packaging format and tools that let you easily deploy the same model (from any ML library) to batch and real-time scoring on platforms such as Docker, Apache Spark, Azure ML and AWS SageMaker.
- MLflow Model Registry: A centralized model store, set of APIs, and UI, to collaboratively manage the full lifecycle of MLflow Models.

In the above data the use of MLflow is very useful for tracking the above model to predict the MAPE, MAE, RMSE and R2 Scores from implementing Linear Regression on this data. Model tracking is also to maintain model consistency if new data is entered

Thank You

Group 3