Internship Report

Artificial Intelligence and Machine Learning Applications

On

"Mobile Price Classification Using Machine Learning"

DLithe Consultancy Services Pvt. Ltd.

Internship Report

Trainee/Intern Name: Kavana P

Reg. no: 4PM22MC103

Period: 6 weeks

Job Assignment:

Organization: DLithe Consultancy Services Pvt. Ltd.

Supervisor's Name: Bhavana A S

Observations:

Submitted to

Signature of Training Supervisor

Date:

Signature of Coordinator

Date:

Letter of Transmittal

To,

Program Co-ordinator

DLithe Consultancy services

Bengaluru

Dear Sir,

I am writing to submit my report on AIML Internship that I recently completed on Artificial Intelligence (AI) and Machine Learning (ML). The training program was an invaluable learning experience, and I am grateful for the opportunity to participate.

The training program covered various aspects of AI and ML, including basic concepts, algorithms, programming languages, and practical applications. I gained a comprehensive understanding of the role of AI and ML in modern technology and industry, and also gained hands-on experience with AI and ML tools and platforms. The training highlighted the potential of AI and ML to revolutionize various fields, including healthcare, finance, and manufacturing.

The report includes a detailed overview of the training program, including the topics covered, the learning objectives, and the outcomes achieved. It also provides observations and insights into the potential benefits and challenges of implementing AI and ML solutions in different fields.

I believe that the knowledge and skills that I acquired during the training program will be valuable to our organization. AI and ML are rapidly becoming more ubiquitous in various industries, and the ability to work with AI and ML tools and platforms will be increasingly important for our organization's success.

I hope that the report provides useful insights into the benefits of on-job training and the potential of AI and ML.

Sincerely,

Name: Kavana P

Reg. no: 4PM22MC013

Table of Contents

Introduction

Artificial Intelligence and Machine Learning are two of the most popular and rapidly growing fields in computer science. They are transforming the way we live, work, and interact with technology. The purpose of this report is to provide an overview of my Internship Training experience on Artificial Intelligence and Machine Learning, and to describe the various concepts and techniques that I learned during the training.

Determining the price of a new mobile phone is a crucial aspect of product strategy in today's diverse and competitive market. Various factors contribute to this decision, requiring a delicate balance to ensure the product's success. Key considerations include the cost of production, encompassing materials, labor, and associated expenses. Research and development (R&D) costs must also be factored in, particularly for phones incorporating innovative features. Market positioning plays a pivotal role, with the perceived value of the product influencing consumer preferences. Competitive pricing analysis is essential, as setting a price significantly higher or lower than competitors can impact market share. Brand image, target audience characteristics, economic conditions, and distribution channels further contribute to the pricing strategy. Additionally, the product's lifecycle stage influences whether an introductory high price or a more competitive one is appropriate. Ultimately, consumers assess the product based on perceived value, considering not only technical specifications but also user experience, design, and brand reputation. Regular market research.

Background

Building a mobile price classifier in Python typically involves using machine learning algorithms to classify mobile phones into price ranges based on certain features like RAM, internal storage, camera quality, etc. Here's an example using a decision tree classifier from the scikit-learn library:

This example assumes you have a dataset where train.csv contains columns for various features like RAM, internal storage, camera quality, etc., and a column named price_range that denotes different price ranges. Before implementing this, ensure you have the necessary dataset formatted properly and that the features are numeric or preprocessed into numeric form. Also, consider feature scaling or normalization if your features are on different scales.

Remember, this is a simple example using a decision tree classifier. You can explore other algorithms (Random Forests, SVC,KNN etc.) and perform hyperparameter tuning to enhance the model's performance. Additionally, feature selection or engineering might improve the classifier's accuracy.

Project Overview

The Internship Training program on Artificial Intelligence and Machine Learning that I participated in was conducted by a technology company. The program was designed to provide a comprehensive overview of the latest advancements in the field of AI and ML, and to equip participants with the skills and knowledge required to build intelligent systems and applications.

The training program consisted of practical hands-on sessions. The lectures covered a wide range of topics, including the fundamentals of AI and ML, various techniques and algorithms used in machine learning, and the latest developments in deep learning and neural networks. The practical sessions involved working on various projects and implementing machine learning algorithms on real-world datasets.

Problem Statement

Mobile prices are an important reflection of the Humans and some ranges are of great interest for both buyers and sellers. Ask a mobile buyer to describe their dream Mobile or Branded Mobile Phones. So in this blog we are going to see about how the prices are segregated based on the some of the features. As well as the target feature prediction based on the same features.

Solution:-

Data Collection and Preprocessing:-

Data Collection:-Gather a dataset containing mobile phone features(RAM, storage, camera etc..) along with their respect prices.

Data Preprocessing:- Clean the data, handle missing values, encoded categorical variables, and scale numerical features if needed.

Model Selection and Training:-

Features Selection/Engineering: Identify important features and transform or create new features if necessary.

Model Selection: Choose a suitable algorithm for classification (e.g.,decision trees, random forest, support vector classifier)

Training: Split the data into training and validation sets. Train the chosen model on the training set.

Methodology

Fig...6.1 Data flow diagram

Data Collection and Pre-processing Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes. The data of Mobile Price is available on Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviours or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues. Data pre-processing prepares raw data for further processing. Given data has 21 feature columns of different 2000 instances. The given data has been cleaned and null values has been removed. Prior to the performing of the machine learning techniques, the data pre-processing was performed on the data set. Incomplete data such as data that which is lacking attribute values, missing values within the records were delete from the data set. Outlier analysis was performed. In WEKA a filter called Interquartile-Range was used to perform outlier analysis.

B. Data Analysis Data Analysis is the process of systematically applying statistical and/or logical techniques to describe and illustrate, condense and recap, and evaluate data. [3] Python's Scikit-Learn Machine Learning Toolbox has been used for the Exploratory Data Analysis, Data Processing and Model Development. Python's Plotting Libraries like Matplotlib and Seaborn have been used for the data Visualizations. Dataset as 21 features and 2000 entries. The meanings of the features are given below.

Features Description

battery_power	Total energy a battery can store in one time measured in mAh
blue	Has bluetooth or not
clock_speed	speed at which microprocessor executes instructions
dual_sim	Has dual sim support or not
fc	Front Camera mega pixels
four_g	Has 4G or not
int_memory	Internal Memory in Gigabytes
m_dep	Mobile Depth in cm
mobile_wt	Weight of mobile phone
n_cores	Number of cores of processor
pc	Primary Camera mega pixels
px_height	Pixel Resolution Height
px_width	Pixel Resolution Width
ram	Random Access Memory in Mega Bytes
sc_h	Screen Height of mobile in cm
sc_w	Screen Width of mobile in cm
talk_time	longest time that a single battery charge will last when you are
three_g	Has 3G or not
touch_screen	Has touch screen or not
wifi	Has wifi or not
price_range	This is the target variable with value of 0(low cost), 1(medium cost), 2(high cost) and 3(very high cost).

Fig. 2: Feature Description of all the columns used.

Fig...6.2 Features description

C. **Correlation Matrix Heat Map**:- A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses.

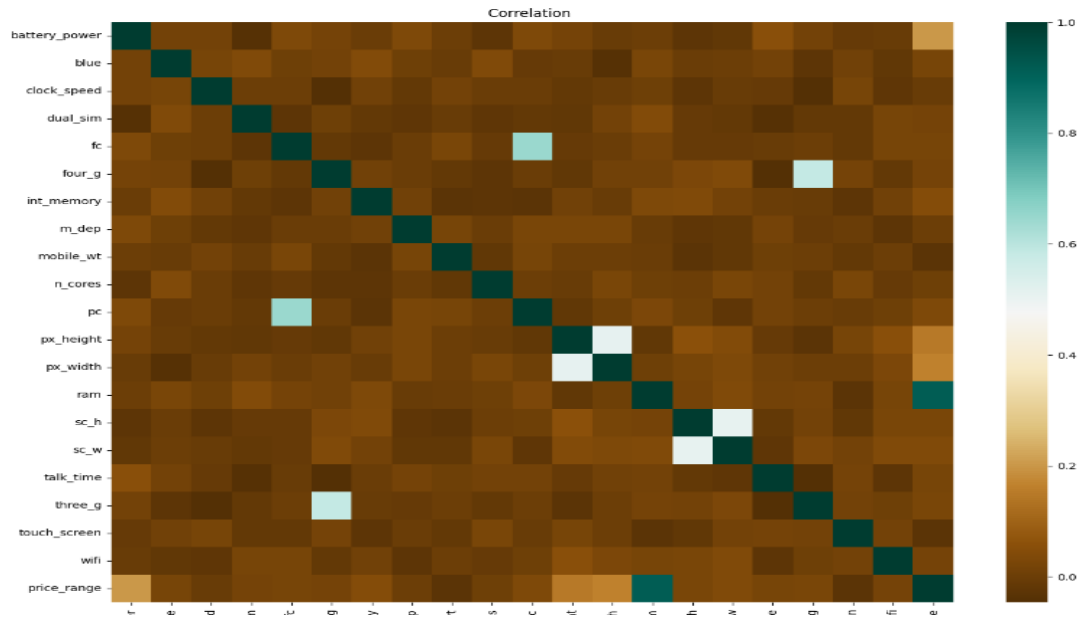

Fig..6.3 Correlation Matrix heap

D.Model Selection and Implementation

1) **Machine Learning**: Machine learning (ML) is categorized under artificial intelligence of (AI) which facilitates the computer with efficiency to perform and learn even after not being particularly programmed. ML is a strategy for information examination that robotizes logical model building.

2) **Classification Algorithms**

a) **Logistic Regression:** Logistic Regression is a supervised learning algorithm used for classification. The model created by this algorithm is based on logistic function . A logistic function, also referred to as a logistic curve is a sigmoid curve with the below equation is the Euler's number, x0 is the x-value of the sigmoid midpoint, M is the curve's maximum value and s is the steepness of the curve.

It is a logistic function to convert the output of a linear regression into classes. Higher linearity between the feature and the target variable contributes to better performance of the Logistic Regression model. In the multiclass case as ours, the training algorithm uses the one-vs-rest (OvR) scheme. The logistic regression class of scikit-learn implements regularized logistic regression. It can handle both dense and sparse input:

```
Train Set Accuracy:64.26666666666667
Test Set Accuracy:63.0

Confusion Matrix:
[[97 24  0  0]
 [27 66 35  1]
 [ 1 30 57 29]
 [ 0  5 33 95]]

Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.80      0.79       121
           1       0.53      0.51      0.52       129
           2       0.46      0.49      0.47       117
           3       0.76      0.71      0.74       133

    accuracy                           0.63       500
   macro avg       0.63      0.63      0.63       500
weighted avg       0.63      0.63      0.63       500
```

Fig...6.4 Logistic Regression Accuracy

b) **K – Nearest Neighbour (KNN):** KNN is a classification algorithm as given in [11] where objects are classified by voting several labelled training examples with their smallest distance from each object. This method performs well even in handling the classification tasks with multi-categorized classification. Its disadvantage is that KNN requires more

time for classifying objects when a large number of training examples are given. KNN should select some of them by computing the distance of each test objects with all of the training examples. KNN is a modest algorithm that stores all accessible suitcases and classifies new suitcases based on a similarity measure. KNN has symmetrical names (a) Memory-Based reasoning (b) Example-Based Reasoning (c) Instance-Based Learning (d) Case-Based Reasoning and (e) Lazy Learning. KNN utilized for relapse and grouping for prescient issues [13]. Be that as it may, it is broadly utilized as a part of grouping troubles in the business.

```
Train Set Accuracy:95.13333333333334
Test Set Accuracy:89.8

Confusion Matrix:
[[124  15   0   0]
 [  1 104  12   0]
 [  0   6 110  14]
 [  0   0   3 111]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.89      0.94       139
           1       0.83      0.89      0.86       117
           2       0.88      0.85      0.86       130
           3       0.89      0.97      0.93       114

    accuracy                           0.90       500
   macro avg       0.90      0.90      0.90       500
weighted avg       0.90      0.90      0.90       500
```

Fig..6.5 K-NN Acuracy

c)**Support Vector Classification (SVC)** :-is a type of supervised learning algorithm within the realm of machine learning used for classification tasks. It's a form of support vector machine (SVM) that aims to find a hyperplane in an N-dimensional space (where N is the number of features) that distinctly classifies the data points.

The key idea behind SVC is to find the optimal hyperplane that maximizes the margin between different classes of data points. This hyperplane is determined by support vectors, which are the data points closest to the decision boundary.

```
Train Set Accuracy:94.93333333333334
Test Set Accuracy:94.0

Confusion Matrix:
[[124   8   0   0]
 [  1 114   9   0]
 [  0   3 112   5]
 [  0   0   4 120]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.94      0.96       132
           1       0.91      0.92      0.92       124
           2       0.90      0.93      0.91       120
           3       0.96      0.97      0.96       124

    accuracy                           0.94       500
   macro avg       0.94      0.94      0.94       500
weighted avg       0.94      0.94      0.94       500
```

Fig..6.6 Support Vector classification

d)**Decision Tree**: A Decision Tree text classifier in [14] is a tree in which internal nodes are labelled by terms, branches departing from them are labelled by the weight that the term has in the text document and leafs are labelled by categories. Decision Tree constructs using 'divide and conquer' strategy. Each node in a tree is associated with set of cases. This strategy checks whether all the training examples have the same label and if not then select a term partitioning from the pooled classes of documents that have same values for term and place each such class in a separate subtree.

```
Train Set Accuracy:100.0
Test Set Accuracy:82.8

Confusion Matrix:
[[112  12   0   0]
 [ 12  99  18   0]
 [  1  14  95  17]
 [  0   0  12 108]]

Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       124
           1       0.79      0.77      0.78       129
           2       0.76      0.75      0.75       127
           3       0.86      0.90      0.88       120

    accuracy                           0.83       500
   macro avg       0.83      0.83      0.83       500
weighted avg       0.83      0.83      0.83       500
```

Fig..6.6 Decision Tree Accuracy

e)AdaBoostClassifier:-

An AdaBoost [1] classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

This class implements the algorithm known as AdaBoost-SAMME

Parameters:-

estimator*object,* default=None

The base estimator from which the boosted ensemble is built. Support for sample weighting is required, as well as proper classes_ and n_classes_ attributes. If None, then the base estimator is **DecisionTreeClassifier** initialized with max_depth=1.

New in version 1.2: base_estimator was renamed to estimator.

n_estimators*int*, default=50

The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early. Values must be in the range [1, inf).

learning_rate*float*, default=1.0

Weight applied to each classifier at each boosting iteration. A higher learning rate increases the contribution of each classifier. There is a trade-off between the learning_rate and n_estimators parameters. Values must be in the range (0.0, inf).

algorithm{'SAMME', 'SAMME.R'}, default='SAMME.R'

If 'SAMME.R' then use the SAMME.R real boosting algorithm. estimator must support calculation of class probabilities. If 'SAMME' then use the SAMME discrete boosting algorithm. The SAMME.R algorithm typically converges faster than SAMME, achieving a lower test error with fewer boosting iterations.

random_stateint, RandomState instance *or None, default=None*

Controls the random seed given at each estimator at each boosting iteration. Thus, it is only used when estimator exposes a random_state. Pass an int for reproducible output across multiple function calls. See Glossary.

base_estimatorobject, default=None

The base estimator from which the boosted ensemble is built. Support for sample weighting is required, as well as proper classes_ and n_classes_ attributes. If None, then the base estimator is **DecisionTreeClassifier** initialized with max_depth=1

```
Train Set Accuracy:100.0
Test Set Accuracy:82.8

Confusion Matrix:
[[112   12    0    0]
 [ 12   99   18    0]
 [  1   14   95   17]
 [  0    0   12  108]]

Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       124
           1       0.79      0.77      0.78       129
           2       0.76      0.75      0.75       127
           3       0.86      0.90      0.88       120

    accuracy                           0.83       500
   macro avg       0.83      0.83      0.83       500
weighted avg       0.83      0.83      0.83       500
```

Fig…6.7 AdaBoost Classifier Accuracy

XGBoost Classifier:-

XGBoost is designed for speed, ease of use, and performance on large datasets. It does not require optimization of the parameters or tuning, which means that it can be used immediately after installation without any further configuration.

XGBoost Features

XGBoost is a widespread implementation of gradient boosting. Let's discuss some features of XGBoost that make it so attractive.

- XGBoost offers regularization, which allows you to control overfitting by introducing L1/L2 penalties on the weights and biases of each tree. This feature is not available in many other implementations of gradient boosting.
- Another feature of XGBoost is its ability to handle sparse data sets using the weighted quantile sketch algorithm. This algorithm allows us to deal with non-zero entries in the feature matrix while retaining the same computational complexity as other algorithms like stochastic gradient descent.
- XGBoost also has a block structure for parallel learning. It makes it easy to scale up on multicore machines or clusters. It also uses cache awareness, which helps reduce memory usage when training models with large datasets.
- Finally, XGBoost offers out-of-core computing capabilities using disk-based data structures instead of in-memory ones during the computation phase.

```
Train Set Accuracy:91.7333333333333
Test Set Accuracy:83.8

Confusion Matrix:
[[114  14   0   0]
 [ 11 103  19   1]
 [  0   8  94  16]
 [  0   0  12 108]]

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.89      0.90       128
           1       0.82      0.77      0.80       134
           2       0.75      0.80      0.77       118
           3       0.86      0.90      0.88       120

    accuracy                           0.84       500
   macro avg       0.84      0.84      0.84       500
weighted avg       0.84      0.84      0.84       500
```

Fig...6.8 XGBoost Accuracy

g)Gradient Boosting for classification.

This algorithm builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage n_classes_ regression trees are fit on the negative gradient of the loss function, e.g. binary or multiclass log loss. Binary classification is a special case where only a single regression tree is induced.

sklearn.ensemble.HistGradientBoostingClassifier

```
Train Set Accuracy:100.0
Test Set Accuracy:82.8

Confusion Matrix:
[[112  12   0   0]
 [ 12  99  18   0]
 [  1  14  95  17]
 [  0   0  12 108]]

Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       124
           1       0.79      0.77      0.78       129
           2       0.76      0.75      0.75       127
           3       0.86      0.90      0.88       120

    accuracy                           0.83       500
   macro avg       0.83      0.83      0.83       500
weighted avg       0.83      0.83      0.83       500
```

Fig…6.9 Gradient Boosting for classification Accuracy

Parameters:-

loss{'log_loss', 'exponential'}, default='log_loss'

The loss function to be optimized. 'log_loss' refers to binomial and multinomial deviance, the same as used in logistic regression. It is a good choice for classification with probabilistic outputs. For loss 'exponential', gradient boosting recovers the **AdaBoost algorithm.**

learning_ratefloat, default=0.1

Learning rate shrinks the contribution of each tree by learning_rate. There is a trade-off between learning_rate and n_estimators. Values must be in the range [0.0, inf).

n_**estimatorsint, default=100**

The number of boosting stages to perform. Gradient boosting is fairly robust to over-fitting so a large number usually results in better performance. Values must be in the range [1, inf).

subsamplefloat, default=1.0

The fraction of samples to be used for fitting the individual base learners. If smaller than 1.0 this results in Stochastic Gradient Boosting. subsample interacts with the parameter n_estimators. Choosing subsample < 1.0 leads to a reduction of variance and an increase in bias. Values must be in the range (0.0, 1.0].

criterion{'friedman_mse', 'squared_error'}, default='friedman_mse'

The function to measure the quality of a split. Supported criteria are 'friedman_mse' for the mean squared error with improvement score by Friedman, 'squared_error' for mean squared error. The default value of 'friedman_mse' is generally the best as it can provide a better approximation in some cases.

New in version 0.18.

min_samples_splitint or float, default=2

The minimum number of samples required to split an internal node:

- If int, values must be in the range [2, inf).
- If float, values must be in the range (0.0, 1.0] and min_samples_split will be ceil(min_samples_split * n_samples).

Changed in vers

Project / Use Case implementation

The objective of the project was to segment credit card customers based on their spending patterns and other attributes. This would help the company to understand the needs and preferences of their customers and develop targeted marketing campaigns and product offerings.

Data Collection: The first step was to collect data from various sources such as credit card transactions, customer demographics, and customer feedback.

Data Preprocessing: The collected data was preprocessed to remove duplicates, missing values, and outliers. Data cleaning techniques such as imputation, normalization, and scaling were applied to prepare the data for analysis.

Feature Selection: The relevant features were selected for analysis based on their importance and correlation with the target variable.

Clustering Algorithm Selection: Different clustering algorithms such as K-Means, Hierarchical Clustering, and DBSCAN were evaluated and compared to select the best algorithm for the data.

Model Training: The selected clustering algorithm was trained on the preprocessed data to generate clusters of credit card customers.

Cluster Analysis: The generated clusters were analyzed and interpreted to gain insights into the spending patterns and preferences of each cluster.

Results Visualization: The results of the analysis were visualized using charts and graphs to make it easy for stakeholders to understand and interpret the findings.

System Requirements:-

- ## Hardware Requirements:-

  Processor **:** IntelCore i3 and Above

  Installed : 8.00GB

  System Type： 64 bit Operating System

- ## Software Requirements:-

  Operating System : Windows 10

  Programming Language: Python

  IDE : Google co lab

  Librariers : Pandas, numpy, matplotlib, seaborn

Google Colab

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.Google Colab makes data science, deep learning, neural network, and machine learning accessible to individual researchers who can not afford costly computational infrastructure. Machine learning and data science are the two new technologies in which all new generation computer scientists want to excel. There are many online learning courses, free lectures, and online how-to guides on ML and data science. importing libraries.

Schematics and Code

```
import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split,GridSearchCV

from sklearn.metrics import accuracy_score,confusion_matrix

import warnings

warnings.filterwarnings('ignore')

#import dataset

df=pd.read_csv('/content/train.csv')

df.head()

df.dtypes

df.info()

df.describe()

df.isnull().sum()
```

```
import missingno as msno

msno.bar(df)

plt.show()

df.var()

df['price_range'].unique()

sns.pointplot(y='ram',x='price_range',data=df)

sns.pointplot(x='price_range',y='battery_power',data=df)

sns.boxplot(x='price_range',y='battery_power',data=df)

sns.pointplot(x='price_range',y='int_memory',data=df)

col = df.columns

categorical_col = ['blue','dual_sim','four_g','three_g','touch_screen','price_range']

for i in categorical_col:

  sns.countplot(df[i])

  plt.xlabel(i)

  plt.show()

for i in df.drop(df[categorical_col],axis=1):

        fig = plt.figure(figsize=(9,8))

        plt.hist(df[i],color='purple',bins=10)

        plt.xlabel(i)

        plt.show()

corr=df.corr()

fig = plt.figure(figsize=(15,12))

r = sns.heatmap(corr, cmap='BrBG')

r.set_title("Correlation ")
```

```
x=df.drop('price_range',axis=1)

y=df['price_range']

scale=StandardScaler()

scaled=scale.fit_transform(x)

from statsmodels.stats.outliers_influence import variance_inflation_factor

vif=pd.DataFrame()

vif['vif']=[variance_inflation_factor(scaled,i) for i in range(scaled.shape[1])]

vif['features']=x.columns

labels = ["3G-supported",'Not supported']

values = df['three_g'].value_counts().values

fig1, ax1 = plt.subplots()

colors = ['red', 'blue']

ax1.pie(values, labels=labels,
autopct='%1.1f%%',shadow=True,startangle=90,colors=colors)

plt.show()

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

lr = LogisticRegression(penalty='l2',C=0.1)

lr.fit(x_train,y_train)

y_test_pred = lr.predict(x_test)

y_train_pred = lr.predict(x_train)

lr_acc=accuracy_score(y_test_pred,y_test)

print("Train Set Accuracy:"+str(accuracy_score(y_train_pred,y_train)*100))

print("Test Set Accuracy:"+str(accuracy_score(y_test_pred,y_test)*100))
```

```
print("\nConfusion Matrix:\n%s"%confusion_matrix(y_test_pred,y_test))

print("\nClassification Report:\n%s"%classification_report(y_test_pred,y_test))

from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n_neighbors=8)

knn.fit(x_train,y_train)

y_test_pred1 = knn.predict(x_test)

y_train_pred1=knn.predict(x_train)

knn_acc=accuracy_score(y_test_pred1,y_test)

print("Train Set Accuracy:"+str(accuracy_score(y_train_pred1,y_train)*100))

print("Test Set Accuracy:"+str(accuracy_score(y_test_pred1,y_test)*100))

print("\nConfusion Matrix:\n%s"%confusion_matrix(y_test_pred1,y_test))

print("\nClassification Report:\n%s"%classification_report(y_test_pred1,y_test))

from sklearn.svm import SVC

svc = SVC()

svc.fit(x_train, y_train)

y_test_pred2 = svc.predict(x_test)

y_train_pred2=svc.predict(x_train)

svc_acc=accuracy_score(y_test_pred2,y_test)

print("Train Set Accuracy:"+str(accuracy_score(y_train_pred2,y_train)*100))

print("Test Set Accuracy:"+str(accuracy_score(y_test_pred2,y_test)*100))

print("\nConfusion Matrix:\n%s"%confusion_matrix(y_test_pred2,y_test))

print("\nClassification Report:\n%s"%classification_report(y_test_pred2,y_test))

from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()
```

```
dtc.fit(x_train, y_train)

y_test_pred3 = dtc.predict(x_test)

y_train_pred3=dtc.predict(x_train)

print("Train Set Accuracy:"+str(accuracy_score(y_train_pred3,y_train)*100))

print("Test Set Accuracy:"+str(accuracy_score(y_test_pred3,y_test)*100))

print("\nConfusion Matrix:\n%s"%confusion_matrix(y_test_pred3,y_test))

print("\nClassification Report:\n%s"%classification_report(y_test_pred3,y_test))

from xgboost import XGBClassifier

xgb = XGBClassifier(booster = 'gbtree', learning_rate = 0.1, max_depth = 5, n_estimators = 10,gamma=5)

xgb.fit(x_train, y_train)

y_test_pred7 = xgb.predict(x_test)

y_train_pred7=xgb.predict(x_train)

xgb_acc= accuracy_score(y_test_pred7,y_test)

print("Train Set Accuracy:"+str(accuracy_score(y_train_pred7,y_train)*100))

print("Test Set Accuracy:"+str(accuracy_score(y_test_pred7,y_test)*100))

print("\nConfusion Matrix:\n%s"%confusion_matrix(y_test_pred7,y_test))

print("\nClassification Report:\n%s"%classification_report(y_test_pred7,y_test))

models = pd.DataFrame({

  'Model': ['Logistic Regression', 'KNN', 'SVC',  'Decision Tree Classifier','Ada Boost Classifier',

        'Gradient Boosting Classifier',  'XgBoost'],

  'Score': [lr_acc, knn_acc, svc_acc, dtc_test_acc, ada_test_acc, gbc_acc, xgb_acc]

})

models.sort_values(by = 'Score', ascending = False)ode
```

```
colors = ["purple", "green", "orange", "magenta","blue","black"]

sns.set_style("whitegrid")

plt.figure(figsize=(16,8))

plt.ylabel("Accuracy %")

plt.xlabel("Algorithms")

sns.barplot(x=models['Model'],y=models['Score'], palette=colors )

plt.show()
```

Results

Comparing the models.

```
models = pd.DataFrame({
    'Model': ['Logistic Regression', 'KNN', 'SVC',  'Decision Tree Classifier','Ada Boost Classifier',
              'Gradient Boosting Classifier',  'XgBoost'],
    'Score': [lr_acc, knn_acc, svc_acc, dtc_test_acc, ada_test_acc, gbc_acc, xgb_acc]
})

models.sort_values(by = 'Score', ascending = False)
```

	Model	Score
2	SVC	0.940
4	Ada Boost Classifier	0.914
5	Gradient Boosting Classifier	0.908
1	KNN	0.898
6	XgBoost	0.838
3	Decision Tree Classifier	0.836
0	Logistic Regression	0.630

Fig..10.1 Comparing the models

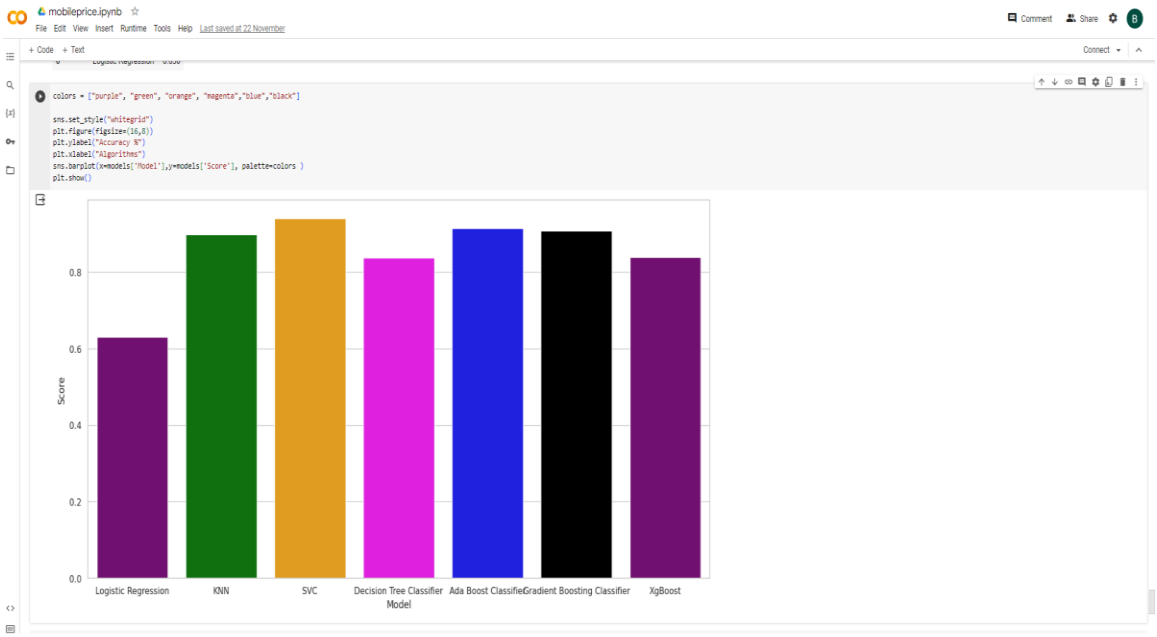

Fig...10.2 Results

Applications of AI and ML:-

- Artificial Intelligence (AI) and Machine Learning (ML) are two of the most promising and rapidly developing technologies today, with numerous potential applications across various fields. Here are some examples of how AI and ML are being applied:
- Healthcare: AI and ML can be used in healthcare to diagnose diseases, analyze medical images and scans, and personalize treatments based on patient data. For example, ML algorithms can be trained to detect cancer in medical images with high accuracy, and AI-powered chatbots can assist patients in diagnosing and treating common illnesses.
- Finance: AI and ML can be used in finance to detect fraud, analyze market trends, and improve risk management. For example, ML algorithms can analyze large amounts of financial data to detect fraudulent transactions, and AI-powered chatbots can assist customers in managing their finances and investments.
- Manufacturing: AI and ML can be used in manufacturing to optimize processes, reduce costs, and improve quality control. For example, ML algorithms can analyze

production data to identify inefficiencies and optimize production processes, and AI-powered robots can be used for repetitive and dangerous tasks.

- Transportation: AI and ML can be used in transportation to improve safety, reduce congestion, and optimize routes. For example, ML algorithms can analyze traffic data to predict and mitigate congestion, and AI-powered vehicles can assist in autonomous driving.

Literature Survey:-

Several articles have studied and investigated machine learning algorithms. First Mark Schmidt focused on SVM and Structural SVM and compared it with logistic regression algorithm with accuracy testing. He found SVM to be more effective compared to Logistic Regression. The paper lack the descriptive learning of all different classification algorithms. Sethi, Kapil & Gupta, Ankit & Gupta, Gaurav & Jaiswal, Varun. (2019). Comparative Analysis of Machine Learning Algorithms on Different Datasets. They have prepared a good research but have focused particular on effect of different size of datasets instead of Different Algorithms. Finally, B. Omar, B. Zineb, A. Cortés Jofré and D. González Cortés, "A Comparative Study of Machine Learning Algorithms for Financial Data Prediction," 2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), Rabat, Morocco, 2018, pp. 1-5, doi: 10.1109/ISAECT.2018.8618774. Have proposed good comparative study but they focused more upon artificial neural networks and other algorithms, they lack the comparison of mostly used algorithms for classification.

- The available and new launching product is an interesting research background for machine learning research.
- Using previous historical data we can predict the price of items by using some machine learning algorithms, best example for this is Sameer Chand-Pudaruth, the researchers worked to estimate the prices of second cars using algorithms such **as** multiple linear regression and k-nearest neighbors (KNN) .

Training Experience

Hands-on Learning: My training program was designed to provide hands-on experience with AI and ML tools and technologies. I was given the opportunity to work on real-world

projects and problems, which helped me develop practical skills and apply theoretical concepts.

Mentorship: I was fortunate to have a mentor who was an experienced AI and ML professional. My mentor provided guidance, feedback, and support throughout my training program, which was invaluable in my learning journey.

Collaboration: One of the most exciting aspects of my training program was the opportunity to work with a team of professionals from different backgrounds. We collaborated on projects and shared ideas, which helped me develop my communication and collaboration skills.

Exposure to Industry Trends: I was able to stay up-to-date with the latest industry trends and developments in AI and ML through various workshops, seminars, and conferences. This helped me gain a broader perspective on the field and prepare for future challenges.

Use of Industry-standard Tools and Technologies: During my training, I had the opportunity to work with industry-standard tools and technologies such as Python, TensorFlow, Keras, and Scikit-Learn. This allowed me to gain practical skills that are in demand in the industry.

Importance of Data Preparation: One of the most important lessons I learned during my training was the critical role of data preparation in the success of AI and ML projects. I learned how to collect, clean, and preprocess data to make it suitable for training models.

Iterative Process: I also learned that developing an AI or ML model is an iterative process that requires a lot of experimentation and tweaking. It is essential to have a feedback loop that allows for continuous improvement of the model.

Key Learnings

During the training program, I learned a range of skills and concepts related to Artificial Intelligence and Machine Learning. Some of the key skills that I acquired are:

Understanding of Artificial Intelligence: I gained a comprehensive understanding of Artificial Intelligence, including the various subfields such as Machine Learning, Deep Learning, and Natural Language Processing.

Machine Learning Concepts and Algorithms: I learned about various Machine Learning concepts and algorithms, including Supervised and Unsupervised Learning, Decision Trees, Random Forests, Support Vector Machines, and K-Nearest Neighbors.

Deep Learning and Neural Networks: I gained a deep understanding of Deep Learning and Neural Networks, including Convolutional Neural Networks and Recurrent Neural Networks.

Programming Skills: I developed strong programming skills in Python, including libraries such as Numpy, Pandas, and Matplotlib.

Data Preprocessing and Analysis: I learned various techniques for data preprocessing and analysis, including Data Cleaning, Data Wrangling, and Exploratory Data Analysis.

Challenges:-

As a technology company that jumped on the AI bandwagon before it became mainstream, we've seen our share of challenged AI projects.

More often than not, artificial intelligence problems stem from a misunderstanding of what AI is, what it is capable of, and whether its implementation makes sense in particular situations.

For example, ever since ChatGPT and other foundation AI models came into prominence, enterprises have been willing to explore generative AI services. Guess how many companies lack an infrastructure for integrating the technology into their processes — and quality data for AI model training.

Essentially, we could boil AI challenges down to five critical issues:

1. Encountering technology-related problems in the development process
2. Failing to reproduce lab results in the real world
3. Struggling to scale AI systems across use cases
4. Making erroneous assumptions about AI capabilities
5. Solving the ethical challenges of AI adoption

Conclusion

The credit card clustering project was a valuable learning experience that allowed me to apply my knowledge of data analytics and machine learning in a real-world setting. I gained hands-on experience with various data preprocessing techniques, clustering algorithms, and visualization tools. I also learned how to interpret and present the results of the analysis to stakeholders. Overall, the project helped me develop practical skills that are in demand in the data analytics industry.

Overall, my Internship Training experience on Artificial Intelligence and Machine Learning was extremely valuable. I gained a solid understanding of the fundamental concepts and techniques in the field, and developed strong programming and data analysis skills. The hands-on projects that I completed during the training gave me a real-world experience of implementing machine learning algorithms on real datasets. I am confident that the skills and knowledge that I acquired during the training will be invaluable in my future career as a data scientist or machine learning engineer.