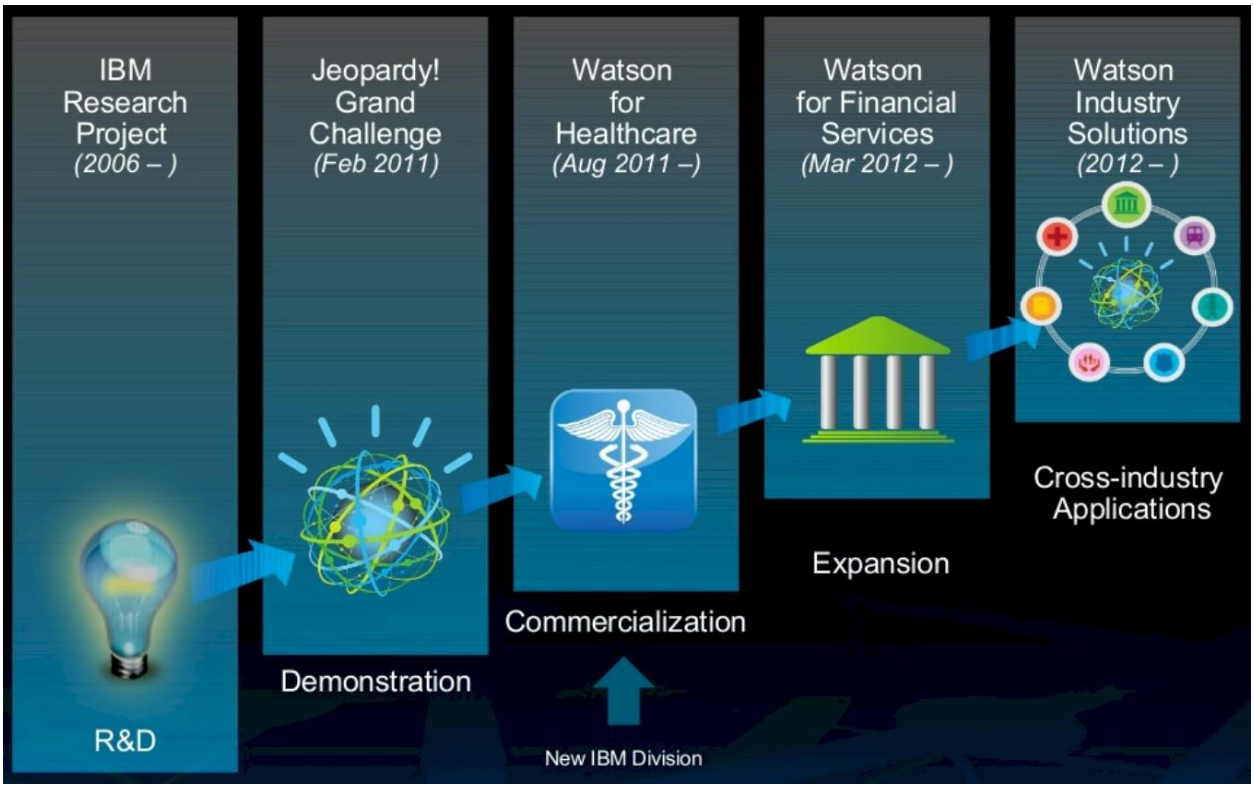
Machine Learning Model Deployment With IBM Cloud Watson Studio

Project Title: ML Models With IBM Watson

Phase 3: Development Part1

Start building the machine learning model using IBM Cloud Watson Studio.



Introduction:

Deploying machine learning models effectively is not just about technological advancements; it's about enhancing human experiences, making processes more efficient, and improving decision-making across various domains. IBM Watson, a frontrunner in artificial intelligence, takes a human-centric approach to modern deployment, aiming to empower individuals, organizations, and communities with intelligent solutions tailored for human use.

In the ever-evolving landscape of data science and artificial intelligence, the ability to develop robust machine learning models is crucial. However, the true value of these models is realized when they are deployed and integrated into real-world applications, enabling businesses to make data-driven decisions and enhance user experiences. IBM Watson offers a comprehensive suite of tools and services that facilitate the seamless deployment of machine learning models, transforming complex algorithms into practical solutions.

The world of machine learning with IBM Cloud Watson Studio! In today's rapidly evolving technological landscape, leveraging the power of data and machine learning is essential for making informed decisions, predicting trends, and solving complex problems. IBM Cloud Watson Studio provides a robust and user-friendly platform that empowers you to build, train, and deploy machine learning models efficiently.

Predictive Use Case:

Customer Churn Prediction :

A predictive use case refers to a specific application of predictive analytics where historical and current data are analyzed to make predictions about future events or outcomes. Predictive analytics involves using statistical algorithms and machine learning techniques to identify patterns, trends, and relationships within data, allowing businesses and organizations to forecast future events accurately.

In a predictive use case:

Historical Data:

Relevant historical data is collected and analyzed. This data typically includes past events, behaviors, transactions, or patterns related to the specific domain of interest.

Features and Target Variable:

The historical data is divided into features (independent variables) and a target variable (dependent variable). Features are the variables used to make predictions, while the target variable is what you want to predict.

Model Training:

Statistical algorithms or machine learning models are trained using the historical data, using features to predict the target variable. During training, the model learns the patterns and relationships in the data.

Prediction:

Once the model is trained and validated, it can be used to make predictions on new, unseen data. The model analyzes the features of the new data to forecast the likely outcome or behavior.

Decision-Making:

Predictions made by the model are used to inform decision-making processes. For example, in customer churn prediction, businesses might use the predictions to identify customers at risk of leaving and implement targeted retention strategies.

Customer Churn:

• Customer churn is defined as when customers or subscribers discontinue doing business with a firm or service.

• Customers in the telecom industry can choose from a variety of service providers and actively switch from one to the next. The telecommunications business has an annual churn rate of 15-25 percent in this highly competitive market.

• Individualized customer retention is tough because most firms have a large number of customers and can't afford to devote much time to each of them. The costs would be too great, outweighing the additional revenue. However, if a corporation could forecast which customers are likely to leave ahead of time, it could focus customer retention efforts only on these "high risk" clients. The ultimate goal is to expand its coverage area and retrieve more customers loyalty. The core to succeed in this market lies in the customer itself.

• Customer churn is a critical metric because it is much less expensive to retain existing customers than it is to acquire new customers.

Given Data Set:

	0	1	2	3	4
customerID	7590-VHVEG	5575-GNVDE	3668-QPYBK	7795-CFOCW	9237-HQITU
gender	Female	Male	Male	Male	Female
SeniorCitizen	0	0	0	0	0
Partner	Yes	No	No	No	No
Dependents	No	No	No	No	No
tenure	1	34	2	45	2
PhoneService	No	Yes	Yes	No	Yes
MultipleLines	No phone service	No	No	No phone service	No
InternetService	DSL	DSL	DSL	DSL	Fiber optic
OnlineSecurity	No	Yes	Yes	Yes	No
OnlineBackup	Yes	No	Yes	No	No
DeviceProtection	No	Yes	No	Yes	No
TechSupport	No	No	No	Yes	No
StreamingTV	No	No	No	No	No
StreamingMovies	No	No	No	No	No
Contract	Month-to-month	One year	Month-to-month	One year	Month-to-month
PaperlessBilling	Yes	No	Yes	No	Yes
PaymentMethod	Electronic check	Mailed check	Mailed check	Bank transfer (automatic)	Electronic check
MonthlyCharges	29.85	56.95	53.85	42.3	70.7
TotalCharges	29.85	1889.5	108.15	1840.75	151.65
Churn	No	No	Yes	No	Yes

Necessary Steps To Follow:

Import Libraries:

Start by importing the necessary libraries:

Program:

```
import pandas as pd
import numpy as np
import missingno as msno
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import warnings
warnings.filterwarnings('ignore')
```

Load the Dataset:

Program:

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import ExtraTreesClassifier
```

```
from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

from xgboost import XGBClassifier

from catboost import CatBoostClassifier

from sklearn import metrics

from sklearn.metrics import roc_curve

from sklearn.metrics import recall_score, confusion_matrix, precision_score, f1_score,
accuracy_score, classification_report

df = pd.read_csv('../input/telco-customer-churn/WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

Data Preprocessing:

Data preprocessing is a crucial step in any machine learning project, including customer churn prediction. It involves transforming raw data into a format that can be effectively used by machine learning models.

Here are the key steps involved in data preprocessing:

1. Data Cleaning.
2. Data Transformation.
3. Data Reduction.
4. Data Splitting.
5. Data Preprocessing Pipeline.

Program:

1. Data Cleaning:

```
df.dropna(inplace=True)

df['column_name'].fillna(df['column_name'].mean(), inplace=True)
```

```
df.drop_duplicates(inplace=True)
```

2. Data Transformation:

```
selected_features = df[['feature1', 'feature2']]
df = pd.get_dummies(df, columns=['categorical_column'])
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
 df['categorical_column'] = le.fit_transform(df['categorical_column'])
from sklearn.preprocessing import StandardScaler, MinMaxScaler
 Standardization
scaler = StandardScaler()
df['numerical_column'] = scaler.fit_transform(df[['numerical_column']])
Min-Max Scaling
scaler = MinMaxScaler()
df['numerical_column'] = scaler.fit_transform(df[['numerical_column']])
```

3. Data Reduction :

```
from sklearn.decomposition import PCA
 PCA for dimensionality reduction
pca = PCA(n_components=2)
reduced_features = pca.fit_transform(features)
```

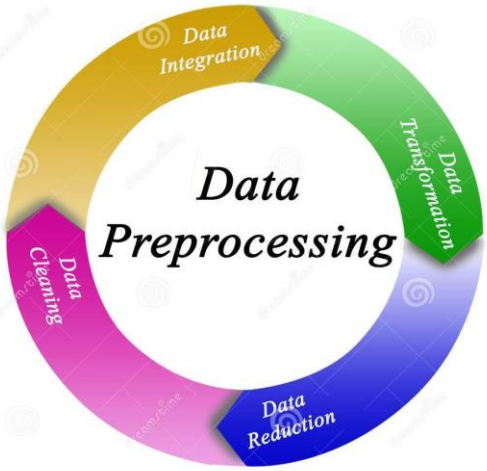
4.Data Splitting:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
```

5. Data Preprocessing Pipeline:

```
from sklearn.pipeline import Pipeline

 Define preprocessing steps

preprocess = Pipeline([

('feature_selection', FeatureSelector()),

('encoding', OneHotEncoder()),

('scaling', StandardScaler())

])

 Apply the preprocessing pipeline to training data

X_train_processed = preprocess.fit_transform(X_train)

 Apply the same preprocessing pipeline to testing data

X_test_processed = preprocess.transform(X_test)
```

After these preprocessing steps, your data is ready to be used for training machine learning models. Remember that the specific preprocessing techniques can vary based on the dataset and the requirements of your machine learning algorithm.



Select Features:

Explore the Data:

Use Watson Studio's data exploration tools to understand your dataset. Visualize data distributions, check for missing values, and understand the characteristics of each feature.

Data Cleaning:

Handle missing values and outliers appropriately. You can use tools within Watson Studio for tasks like imputation and outlier detection.

Feature Selection:

Based on your exploration, select relevant features for your model. For example, 'age', 'monthly_spend', and 'customer_service_calls' might be important features for churn prediction.

Train the Machine Learning Model:

splitting the data into features (X) and the target variable (y), and splitting the data into training and testing sets.

Training a machine learning model refers to the process of teaching the model to recognize patterns in the input data (features) and associate them with corresponding outputs (labels or predictions). During the training process, the model learns from historical data so that it can make accurate predictions or decisions when presented with new, unseen data.

Splitting the data into train and test sets

```
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, classification_report

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

model = LogisticRegression()

model.fit(X_train, y_train)
```

```
predictions = model.predict(X_test)

accuracy = accuracy_score(y_test, predictions)

print(f'Accuracy: {accuracy:.2f}')

print(classification_report(y_test, predictions))
```

Conclusion:

Machine learning model deployment bridges the gap between theoretical analysis and practical applications. By deploying models, businesses can harness predictive capabilities to automate decision-making processes, enhance user experiences, optimize operations, and gain a competitive edge. Efficient deployment ensures that the models are seamlessly integrated into production systems, enabling organizations to transform raw data into actionable intelligence, thereby driving innovation and delivering significant value to stakeholders and end-users.

We've now set up a basic machine learning model using IBM Cloud Watson Studio. We can further customize and improve the machine learning modern deployment functionality by expanding the features.