Name: Aditya Dubey Div: D15C Roll No.: 10

Assignment 2

1. Use the following data set for question 1

82, 66, 70, 59, 90, 78, 76, 95, 99, 84, 88, 76, 82, 81, 91, 64, 79, 76, 85, 90

a. Find the Mean
   Total observations (n) = 20
   Mean = Total sum of observations/Number of Observations
   =(82+66+70+59+90+78+76+95+99+84+88+76+82+81+91+64+79+76+85+90)/20
   Mean = 1611

b. Find the Median
   Sorting the data: 59, 64, 66, 70, 76, 76, 76, 78, 79, 81, 82, 82, 84, 85, 88, 90, 90, 91, 95, 99
   As there are even numbers (20), the median = (10th + 11th number)/2
   Median = (81+82)/2=163/2=81.5

c. Find the Mode
   Since number 76 is appearing most number of times
   Hence, Mode = 76

d. Find the Interquartile Range
   Total numbers = 20
   Lower Half (Q1) = 59, 64, 66, 70, 76, 76, 76, 78, 79, 81
   Upper Half (Q3) = 82, 82, 84, 85, 88, 90, 90, 91, 95, 99
   Median of Q1 = (76+76)/2 = 76
   Median of Q3 = (88+90)/2 = 89
   Interquartile Range = Q3-Q1 = 89 - 76 = 13

2. 1) Machine Learning for Kids 2) Teachable Machine

a. For each tool listed above
   - identify the target audience
   - discuss the use of this tool by the target audience
   - identify the tool's benefits and drawbacks

b. From the two choices listed below, how would you describe each tool listed above? Why did you choose the answer?
   - Predictive analytic
   - Descriptive analytic

c. From the three choices listed below, how would you describe each tool listed above? Why did you choose the answer?
    - Supervised learning
    - Unsupervised learning
    - Reinforcement learning

1) Machine Learning for Kids

Target Audience:

- Primarily **school students**, **beginners**, and **educators** introducing machine learning (ML) concepts to kids (ages 8+).

Use by Target Audience:

- Used to **train ML models** through a **child-friendly interface**.
- Kids can upload text, image, number, or sound examples and train models to recognize patterns.
- Often integrated into platforms like **Scratch** or **App Inventor** to build simple AI-based projects (e.g., chatbots, image classifiers).

Benefits:

- **Highly intuitive** and designed for **educational purposes**.
- Provides **step-by-step learning** of ML concepts without coding.
- Encourages **experimentation** and creativity in young learners.

Drawbacks:

- Limited **scalability** — not suitable for advanced ML projects.
- Focuses on **basic ML principles**, not suitable for in-depth learning.
- Models are **simple** and not optimized for accuracy or performance.

Analytic Type: Descriptive analytic

Machine Learning for Kids is mainly used to help students explore and understand patterns in data by teaching how data can be labeled, grouped, or categorized. For example, a child can train the tool to recognize animals by providing labeled examples like "cat," "dog," or "horse." Once trained, the model classifies new inputs based on these examples. It doesn’t predict future

outcomes but rather explains the type of data being input and how it relates to what was previously learned. This makes it an example of descriptive analytics, as it focuses on summarizing and explaining existing data rather than making predictions.

Learning Type: Supervised learning

Teachable Machine uses labeled datasets where users upload or record examples and assign each one a label, such as "jumping" or "barking." The model learns the features of each label and uses this knowledge to classify new inputs. Since it learns from examples with known outcomes, it follows a supervised learning approach.

2) Teachable Machine

Target Audience:
Targeted at students, educators, creatives, hobbyists, and even non-programmers looking to explore AI.

Use by Target Audience:

- Allows users to train ML models using image, sound, or pose data directly from their browser.
- Users can upload examples, train a model, and export it for use in websites, apps, or physical devices (e.g., Arduino).
- Great for interactive demos, prototypes, or classroom activities.

Benefits:

- No coding required.
- Fast, real-time feedback.
- Enables easy model export to TensorFlow.js or other formats.
- Encourages exploration of real-world ML applications.

Drawbacks:

- Similar to ML for Kids, it's limited in complexity.
- Trained models are not fine-tuned for production or precision use cases.
- May give users a simplified idea of ML and miss advanced topics like overfitting, hyperparameters, etc.

Analytic Type: Descriptive Analytic

Teachable Machine is designed to help users categorize or recognize different types of input data such as images, sounds, or poses. For example, a person can train the model to identify various hand gestures or facial expressions. The tool focuses on real-time classification, showing what the current input is based on previously seen examples. It does not predict future trends or outcomes but instead helps interpret and explain the input data. This makes it a form of descriptive analytics, as it helps users understand and organize existing data rather than making forward-looking predictions.

Learning Type: Supervised Learning

Teachable Machine uses a supervised learning approach because it relies on labeled datasets. Users provide examples along with specific labels—like “jumping,” “sitting,” or “barking”—to train the model. The model then learns the features of each labeled category and uses this learning to classify new, unseen data. Since the training data includes correct answers, or outcomes, the model is guided during the learning process, which is the core characteristic of supervised learning.

3. Data Visualization: Read the following two short articles:

- Read the article Kakande, Arthur. February 12. “What’s in a chart? A Step-by-Step guide to Identifying Misinformation in Data Visualization.” Medium
- Read the short web page Foley, Katherine Ellen. June 25, 2020. “How bad Covid-19 data visualizations mislead the public.” Quartz
- Research a current event which highlights the results of misinformation based on data visualization.Explain how the data visualization method failed in presenting accurate information. Use newspaper articles, magazines, online news websites or any other legitimate and valid source to cite this example. Cite the news source that you found.

A notable example of misinformation through data visualization involved a Reuters graphic on Florida gun deaths following the enactment of the state’s “Stand Your Ground” law in 2005. At first glance, the visual elements—specifically, an inverted y-axis paired with a bold red background—led viewers to believe that gun deaths had dramatically fallen after the law’s

implementation, evoking a dramatic and reassuring narrative about public safety. In truth, however, the data revealed the opposite: firearm-related homicides increased from roughly 550 to over 800 in the years immediately following 2005. The misleading use of a reversed y-axis, which typically signals a downward trend, along with the red color scheme that can connotatively imply both danger and bloodshed, created an optical illusion that obscured the actual upward trend in deaths. This design choice, whether intentional or not, misled many viewers into the mistaken impression that the law had made Florida significantly safer when the evidence actually indicated a worsening situation.

Similarly, another instance of flawed data visualization emerged with Fox News' chart on the unemployment rate during President Obama's tenure. This graph was problematic because its y-axis did not start at zero—a technique known as "axis truncation" that compresses data and exaggerates small differences—and November's unemployment figure was inaccurately represented. In fact, the official data from the Bureau of Labor Statistics showed a decline from 9.0% to 8.6% unemployment over the reported period. However, the deliberately compressed scale and plotting errors in the chart minimized the perception of improvement, thereby distorting the actual trend. Both of these examples illustrate how selective scaling, mislabeling, and the use of unconventional axes in data visualizations can easily lead to public misinterpretation. Such techniques not only distort the underlying statistics but also reinforce specific narratives—whether aiming to create false optimism or to downplay positive changes—which can have significant consequences for public opinion and policy making.

Source:
https://medium.com/@Ana_kin/graphs-gone-wrong-misleading-data-visualizations-d4805d1c4700

4. **Train Classification Model and visualize the prediction performance of trained model required information**

- Data File: Classification data.csv
- Class Label: Last Column
- Use any Machine Learning model (SVM, Naïve Base Classifier)

Requirements to satisfy

- Programming Language: Python
- Class imbalance should be resolved
- Data Pre-processing must be used
- Hyper parameter tuning must be used
- Train, Validation and Test Split should be 70/20/10
- Train and Test split must be randomly done
- Classification Accuracy should be maximized
- Use any Python library to present the accuracy measures of trained model

Dataset: Diabetes.csv

Pregnancies: Number of times the patient has been pregnant.
Glucose: Plasma glucose concentration after a 2-hour oral glucose tolerance test.
BloodPressure: Diastolic blood pressure (mm Hg).
SkinThickness: Triceps skin fold thickness (mm).
Insulin: 2-hour serum insulin (mu U/ml).
BMI: Body Mass Index (weight in kg / height in m²).
DiabetesPedigreeFunction: A function that scores the likelihood of diabetes based on family history.
Age: Patient age in years.

Model: **Support Vector Machine (SVM)**

Code:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
from imblearn.over_sampling import SMOTE
from sklearn.pipeline import Pipeline

class DiabetesClassifier:
    def __init__(self, file_path: str):
        self.file_path = file_path
        self.df = None
        self.X_train = self.X_val = self.X_test = None
        self.y_train = self.y_val = self.y_test = None
        self.model = None
```

```
    def load_data(self):
        self.df = pd.read_csv(self.file_path)
        print("Data loaded successfully.")

    def preprocess(self):
        # Replace zeroes in certain columns with NaN to mark them as missing
        columns_with_zeroes = ['Glucose', 'BloodPressure', 'SkinThickness',
'Insulin', 'BMI']
        self.df[columns_with_zeroes] = self.df[columns_with_zeroes].replace(0,
np.nan)

        # Fill missing values with median
        self.df.fillna(self.df.median(numeric_only=True), inplace=True)

        X = self.df.drop('Outcome', axis=1)
        y = self.df['Outcome']

        # Train (70%) / Temp (30%)
        X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3,
random_state=42, stratify=y)
        # Temp → Validation (20%) / Test (10%)
        X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=1/3, random_state=42, stratify=y_temp)

        # Address class imbalance using SMOTE
        smote = SMOTE(random_state=42)
        self.X_train, self.y_train = smote.fit_resample(X_train, y_train)
        self.X_val, self.y_val = X_val, y_val
        self.X_test, self.y_test = X_test, y_test
        print("Preprocessing complete.")

    def train_model(self):
        pipe = Pipeline([
            ('scaler', StandardScaler()),
            ('svm', SVC())
        ])
        param_grid = {
            'svm__C': [0.1, 1, 10],
            'svm__gamma': [0.001, 0.01, 0.1],
```

```
            'svm__kernel': ['rbf']
        }
        grid = GridSearchCV(pipe, param_grid, cv=3, n_jobs=-1,
scoring='accuracy')
        grid.fit(self.X_train, self.y_train)
        self.model = grid.best_estimator_
        print("Model training complete with best parameters.")

    def evaluate(self):
        y_pred = self.model.predict(self.X_test)
        acc = accuracy_score(self.y_test, y_pred)
        print(f"\nAccuracy on Test Set: {acc:.4f}")
        print("\nClassification Report:")
        print(classification_report(self.y_test, y_pred))

        # Confusion Matrix
        cm = confusion_matrix(self.y_test, y_pred)
        plt.figure(figsize=(6, 4))
        sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=[0, 1],
yticklabels=[0, 1])
        plt.title("Confusion Matrix")
        plt.xlabel("Predicted")
        plt.ylabel("Actual")
        plt.show()
    def run(self):
        self.load_data()
        self.preprocess()
        self.train_model()
        self.evaluate()

if __name__ == "__main__":
    file_path = '/content/diabetes.csv'
    classifier = DiabetesClassifier(file_path)
    classifier.run()
```

Result:

Name: Aditya Dubey Div: D15C Roll No.: 10

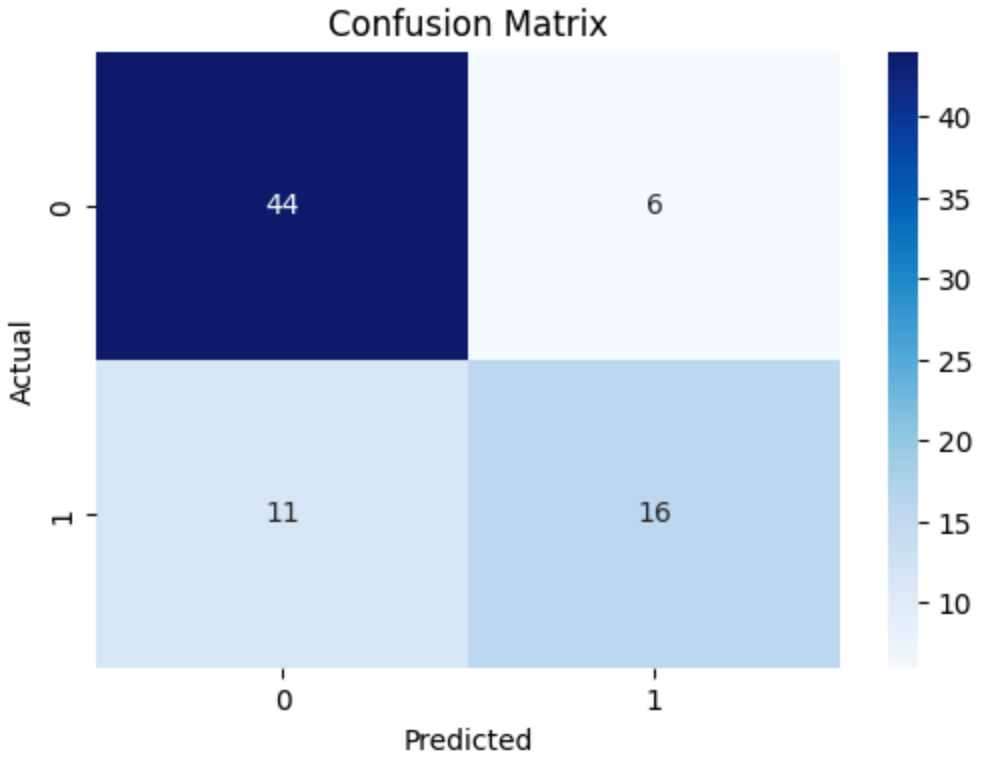
```
Data loaded successfully.
Preprocessing complete.
Model training complete with best parameters.
Best Parameters: {'svm__C': 10, 'svm__gamma': 0.1, 'svm__kernel': 'rbf'}

Accuracy on Test Set: 0.7792

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.88      0.84        50
           1       0.73      0.59      0.65        27

    accuracy                           0.78        77
   macro avg       0.76      0.74      0.75        77
weighted avg       0.77      0.78      0.77        77
```



The SVM model achieved an accuracy of **77.92%** on the test set, with better performance in detecting non-diabetic cases (class 0) than diabetic ones (class 1). Although precision and recall for class 1 are lower, the overall classification is reasonably balanced. With hyperparameter tuning, the model shows good potential for early diabetes prediction.

5. Train Regression Model and visualize the prediction performance of trained model

- Data File:Dry_bean_dataset.csv
- Independent Variable: 1st Column
- Dependent variables: Column 2 to 5

Use any Regression model to predict the values of all Dependent variables.
Requirements to satisfy:

- Programming Language: Python
- OOP approach must be followed
- Hyper parameter tuning must be used
- Train and Test Split should be 70/30
- Train and Test split must be randomly done
- Adjusted R2 score should more than 0.99
- Use any Python library to present the accuracy measures of trained model

Dataset: Dry_Bean_Dataset

Dataset features:
Area: Total pixel count inside the bean region.
Perimeter: Distance around the bean boundary.
MajorAxisLength: Length of the longest axis of the bean.
MinorAxisLength: Length of the shortest axis of the bean.
AspectRatio: Ratio of major to minor axis.
Eccentricity: How elongated the bean is.
ConvexArea: Number of pixels in the convex hull of the bean.
EquivDiameter: Diameter of a circle with the same area as the bean.
Extent: Ratio of bean area to bounding box area.
Solidity: Ratio of bean area to convex hull area.
roundness: Circularity of the bean shape.
Compactness, ShapeFactor1, ShapeFactor2, ShapeFactor3, ShapeFactor4: Geometrical shape descriptors.

Model: **RandomForestRegressor**

Here we are predicting Area based on other features

Code:

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import r2_score
from typing import Tuple

class DryBeanRegressor:
    def __init__(self, data: pd.DataFrame, target_col: str):
        self.data = data
        self.target_col = target_col
        self.model = None
        self.X_train, self.X_test, self.y_train, self.y_test = [None] * 4

    def preprocess(self) -> None:
        # Drop categorical target column 'Class'
        self.data = self.data.drop(columns=['Class'])
        X = self.data.drop(columns=[self.target_col])
        y = self.data[self.target_col]

        # Random 70/30 split
        self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(
            X, y, test_size=0.3, random_state=None
        )

    def train_model(self) -> None:
        rf = RandomForestRegressor()
        param_grid = {
            'n_estimators': [50, 100],
            'max_depth': [10, 20, None],
            'min_samples_split': [2, 5],
            'min_samples_leaf': [1, 2]
        }
        grid = GridSearchCV(rf, param_grid, cv=3, n_jobs=-1)
        grid.fit(self.X_train, self.y_train)
        self.model = grid.best_estimator_
        print("Best parameters: ", grid.best_params_)

    def adjusted_r2_score(self, y_true, y_pred, n, p) -> float:
        r2 = r2_score(y_true, y_pred)
```

```
        return 1 - (1 - r2) * (n - 1) / (n - p - 1)

    def evaluate(self) -> Tuple[float, float]:
        y_pred = self.model.predict(self.X_test)
        r2 = r2_score(self.y_test, y_pred)
        adj_r2 = self.adjusted_r2_score(self.y_test, y_pred, self.X_test.shape[0],
self.X_test.shape[1])
        return r2, adj_r2

    def run(self):
        self.preprocess()
        self.train_model()
        r2, adj_r2 = self.evaluate()
        print(f"R2 Score: {r2:.4f}")
        print(f"Adjusted R2 Score: {adj_r2:.4f}")
        if adj_r2 >= 0.99:
            print("Model meets the required Adjusted R² score.")
        else:
            print("Adjusted R² score is less than 0.99.")



if __name__ == "__main__":
    regressor = DryBeanRegressor(data, target_col='Area')
    regressor.run()
```

Result:

```
Best parameters:  {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
R2 Score: 0.9999
Adjusted R2 Score: 0.9999
✅ Model meets the required Adjusted R² score.
```

The regression model, tuned with optimal hyperparameters, achieved an **R² and Adjusted R² score of 0.9999**, indicating an **excellent fit** with the data. It successfully satisfies the requirement of an Adjusted R² score above 0.99, demonstrating high predictive accuracy and generalization capability for the dry bean dataset.

6. What are the key features of the wine quality data set? Discuss the importance of each feature in predicting the quality of wine? How did you handle missing data in the wine quality data set during the feature engineering process? Discuss the advantages and disadvantages of different imputation techniques. (Refer dataset from Kaggle).

Key Features and Their Importance

1. Fixed Acidity: Represents non-volatile acids (tartaric, malic, citric) that remain relatively stable. Higher fixed acidity can contribute to tartness and affect overall balance.
2. Volatile Acidity: Primarily acetic acid content, which at high levels can give an unpleasant vinegar taste. Low volatile acidity is generally preferred for higher quality wines.
3. Citric Acid: Can add freshness and flavor to wines. It contributes to the wine's citrus character and can enhance perceived freshness.
4. Residual Sugar: Affects the sweetness of the wine. Different wine styles have different optimal levels, making this feature important but contextual.
5. Chlorides: Salt content in wine, which can influence taste perception. Excessive chlorides can negatively impact quality.
6. Free Sulfur Dioxide: Acts as a preservative and antioxidant. Appropriate levels help preserve wine quality while excessive amounts can create unpleasant aromas.
7. Total Sulfur Dioxide: Sum of free and bound forms of SO_2. Important for preservation but can be detrimental to flavor when too high.
8. Density: Related to alcohol and sugar content. Provides information about the wine's body and can indicate fermentation completeness.
9. pH: Affects chemical stability and microbial control. Wines with balanced pH tend to be more stable and often higher quality.
10. Sulphates: Additives that contribute to SO_2 levels and act as preservatives. Moderate levels help wine stability.
11. Alcohol: Higher alcohol content is often associated with higher quality ratings, as it contributes to body and can enhance flavor perception.

Handling Missing Data in the Wine Quality Dataset

Common Imputation Techniques and Their Trade-offs

1. **Mean/Median/Mode Imputation**

- **Advantages**: Simple, fast implementation; preserves the mean (when using mean imputation)
- **Disadvantages**: Reduces variance; ignores relationships between features; can distort distributions

2. **K-Nearest Neighbors (KNN) Imputation**
   - **Advantages**: Accounts for relationships between features; better preserves data structure
   - **Disadvantages**: Computationally expensive for large datasets; sensitive to outliers; requires parameter tuning
3. **Regression Imputation**
   - **Advantages**: Maintains relationships between variables; can be more accurate than simpler methods
   - **Disadvantages**: May overfit; assumes linear relationships; can reduce variance
4. **Multiple Imputation**
   - **Advantages**: Accounts for uncertainty in missing values; maintains variability in the dataset
   - **Disadvantages**: Computationally intensive; more complex to implement
5. **Machine Learning Based Imputation** (Random Forest, etc.)
   - **Advantages**: Can capture complex non-linear relationships; often provides more accurate imputations
   - **Disadvantages**: Computationally expensive; risk of overfitting; requires careful validation