Jupyter Notebook Analysis

```
#Importing neccesary libaries

import pandas as pd

import numpy as np
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
#Reading the csv file with pandas

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

#Shows the first 5 rows

df.head()
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
#Separating the independent variables (features)

X = df.drop(columns='Purchased')


#Setting the dependent variable (target)

y = df['Purchased']


#Converting the independent variables (features) into a pandas DataFrame

df_X = pd.DataFrame(X)

#Converting the dependent variable (target) into a pandas DataFrame

df_y = pd.DataFrame(y)

#Displaying the first 5 rows of the independent data (features)

print("Our independent data (Features):")

print(df_X.head())

#Displaying the first 5 rows of the dependent data (target)
```

```
print("\nOur dependent data (Target):")
print(df_y.head())
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
from sklearn.model_selection import train_test_split
#Splittign dataset to train and test %20 for testing %80 for training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2,
random_state=42)
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
from sklearn.preprocessing import StandardScaler
#It scales data to have a mean of 0 and a standart deviation of 1
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
from sklearn.svm import SVC
svc = SVC()
#The model learns patterns from the standardized training features (X_train) and their
corresponding labels (y_train)
svc.fit(X_train, y_train)
#Making predictions using test data by trained model
svc_ypred = svc.predict(X_test)
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn= KNeighborsClassifier()
#The model memorizes the training data points (X_train) and their corresponding labels
(y_train)
knn.fit(X_train, y_train)
#Making predictions using the test data by trained model
knn_ypred = knn.predict(X_test)
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
from sklearn.metrics import confusion_matrix
#Calculating the confusion matrix of the SVC and KNN predictions
svc_matrix = confusion_matrix(y_test, svc_ypred)
knn_matrix = confusion_matrix(y_test, knn_ypred)


# Displaying the confusion matrix for the SVC model.
print("Confusion Matrix for SVC:")
print(f"True Positive: {svc_matrix[1, 1]}  |  False Positive: {svc_matrix[0, 1]}")
print(f"False Negative: {svc_matrix[1, 0]}  |  True Negative: {svc_matrix[0, 0]}\n")
# Displaying the confusion matrix for the KNN model.
print("Confusion Matrix for KNN:")
print(f"True Positive: {knn_matrix[1, 1]}  |  False Positive: {knn_matrix[0, 1]}")
print(f"False Negative: {knn_matrix[1, 0]}  |  True Negative: {knn_matrix[0, 0]}")
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
#Creating copy of the original features dataset
X_new = X.copy()
#Randomly selecting 100 indices from the dataset without replacement
nan_x = np.random.choice(X_new.index, size=100, replace=False)
```

```
#Setting the selected rows NaN

X_new.loc[nan_x] = np.nan

#Show the first 60 rows

X_new.head(60)
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
#Checking for tatol missing values for columns in dataset

X_new.isnull().sum()
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
#Replacing 'Age' column with mean of this column

X_new['Age'] = X_new['Age'].replace(np.nan, X['Age'].mean())


#Replacing 'EstimatedSalary' column with mean of this column

X_new['EstimatedSalary']=                        X_new['EstimatedSalary'].replace(np.nan,

X['EstimatedSalary'].mean())
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
#Checking if our nan values properly replaced by the mean

X_new.isnull().sum()
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
#Show the first 50 rows

X_new.head(50)
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
# Splitting the dataset into training and testing sets with 80% for training and 20% for
```

```
testing.
# Using a new variable set for the split: X_train2, X_test2, y_train2, y_test2.
X_train2, X_test2, y_train2, y_test2 = train_test_split(X_new, y, test_size=0.2,
random_state=42)
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
# Creating a new StandardScaler object (sc2) because the feature values were modified.
# Re-scaling the new training and testing sets to standardize the features.
sc2 = StandardScaler()
# Fit and transform the new training data.
X_train2 = sc2.fit_transform(X_train2)


# Transform the new testing data using the same scaler.
X_test2 = sc2.transform(X_test2)
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
# Creating and training an SVM classifier (SVC) on the scaled training data.
# The model is fitted to the training features (X_train2) and corresponding labels
(y_train2).
svc2 = SVC()
svc2.fit(X_train2, y_train2)
# Fit the SVC model to the training data.


svc2_ypred = svc2.predict(X_test2)
# Predict the labels for the test data.
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
# Creating and training a K-Nearest Neighbors classifier (KNeighborsClassifier) on the scaled training data.
# The model is fitted to the training features (X_train2) and corresponding labels (y_train2).
knn2 = KNeighborsClassifier()
knn2.fit(X_train2, y_train2)
# Fit the KNN model to the training data.


knn2_ypred = knn2.predict(X_test2)
# Predict the labels for the test data.
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

```
# Calculating confusion matrices for both the SVC and KNN models.
svc2_matrix = confusion_matrix(y_test2, svc2_ypred)   # Confusion matrix for the SVC model.
knn2_matrix = confusion_matrix(y_test2, knn2_ypred)   # Confusion matrix for the KNN model.


# Displaying the confusion matrix for the SVC model.
print("Confusion Matrix for SVC:")
print(f"True Positive: {svc2_matrix[1, 1]}  |  False Positive: {svc2_matrix[0, 1]}")
print(f"False Negative: {svc2_matrix[1, 0]}  |  True Negative: {svc2_matrix[0, 0]}\n")


# Displaying the confusion matrix for the KNN model.
print("Confusion Matrix for KNN:")
print(f"True Positive: {knn2_matrix[1, 1]}  |  False Positive: {knn2_matrix[0, 1]}")
print(f"False Negative: {knn2_matrix[1, 0]}  |  True Negative: {knn2_matrix[0, 0]}")
```

Explanation: This section executes the provided code. Analyze the purpose and outputs.

Explanation: This section executes the provided code. Analyze the purpose and outputs.