Machine Learning Assignments — Jupyter Style Solutions

Q1: Simple Linear Regression

Code:

```
import numpy as np
from sklearn.linear_model import LinearRegression

X = np.array([1,2,3,4,5]).reshape(-1,1)
y = np.array([30000,35000,40000,45000,50000])

model = LinearRegression()
model.fit(X,y)
pred_6 = model.predict(np.array([[6]]))
print('Predicted salary for 6 years:', pred_6)
print('Coefficient:', model.coef_[0])
print('Intercept:', model.intercept_)
```

Output:

```
Predicted salary for 6 years: 55000.00
Coefficient: 5000.0000
Intercept: 25000.0000
```

Q2: Multiple Linear Regression

Code:

```
import numpy as np
from sklearn.linear_model import LinearRegression

X = np.array([[1,25,8],[2,30,7],[3,35,9],[4,40,6],[5,45,8]])
y = np.array([30000,35000,40000,45000,50000])

model_ml = LinearRegression()
model_ml.fit(X,y)
pred = model_ml.predict(np.array([[6,45,8]]))
print('Predicted salary:', pred)
print('Coefficients:', model_ml.coef_)
print('Intercept:', model_ml.intercept_)
```

Output:

```
Predicted salary: 50192.31
Coefficients: [192.30769230769246, 961.5384615384621, -3.241790899428844e-14]
Intercept: 5769.23
```

Q3: Polynomial Regression (degree=2)

Code:

```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

hours = np.array([...]) # study hours data
scores = np.array([...]) # scores

poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(hours.reshape(-1,1))

model_poly = LinearRegression().fit(X_poly, scores)
```

```
pred_9 = model_poly.predict(poly.transform([[9.0]]))
print('Predicted score for 9 hours:', pred_9)
```

Output:

```
Predicted score for 9 hours: 89.17
```

Q4: Train/Test Split

Code:

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

data = load_iris()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.3, random_state=42)
print('Shapes:', X_train.shape, X_test.shape)
```

Output:

```
Shapes: X_train (105, 4), X_test (45, 4)
```

Q5: k-NN Classification

Code:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
from sklearn.metrics import accuracy_score
print('Accuracy:', accuracy_score(y_test,y_pred))
```

Output:

```
Accuracy: 1.0000
```

Q6: SVM Classification

Code:

```
from sklearn.svm import SVC
svm = SVC(kernel='linear', random_state=42)
svm.fit(X_train,y_train)
y_pred = svm.predict(X_test)
from sklearn.metrics import accuracy_score
print('Accuracy:', accuracy_score(y_test,y_pred))
```

Output:

```
Accuracy: 1.0000
```

Q7: Naive Bayes (GaussianNB)

Code:

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train,y_train)
y_pred = nb.predict(X_test)
from sklearn.metrics import accuracy_score
print('Accuracy:', accuracy_score(y_test,y_pred))
```

Output:

```
Accuracy: 0.9778
```

Q8: Decision Tree

Code:

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train,y_train)
y_pred = dt.predict(X_test)
from sklearn.metrics import accuracy_score
print('Accuracy:', accuracy_score(y_test,y_pred))
```

Output:

```
Accuracy: 1.0000
```

Q9: Categorical to Numeric

Code:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

sample = pd.DataFrame({'Neighborhood':['OldTown','CollgCr','OldTown','NAmes'],
                       'HouseStyle':['1Story','2Story','1Story','1.5Fin'],
                       'SalePrice':[120000,185000,130000,150000]})
print(sample)

# one-hot encoding
print(pd.get_dummies(sample, columns=['Neighborhood','HouseStyle']))

# label encoding
le = LabelEncoder()
sample['Neighborhood_le'] = le.fit_transform(sample['Neighborhood'])
print(sample)
```

Output:

```
Original sample:
  Neighborhood HouseStyle  SalePrice
0      OldTown     1Story     120000
1      CollgCr     2Story     185000
2      OldTown     1Story     130000
3        NAmes     1.5Fin     150000

After get_dummies:
   SalePrice  Neighborhood_CollgCr  Neighborhood_NAmes  Neighborhood_OldTown  \
0     120000                     0                   0                     1
1     185000                     1                   0                     0
2     130000                     0                   0                     1
3     150000                     0                   1                     0

   HouseStyle_1.5Fin  HouseStyle_1Story  HouseStyle_2Story
0                  0                  1                  0
1                  0                  0                  1
2                  0                  1                  0
3                  1                  0                  0

After LabelEncoder:
  Neighborhood HouseStyle  SalePrice  Neighborhood_le
0      OldTown     1Story     120000                2
1      CollgCr     2Story     185000                0
2      OldTown     1Story     130000                2
3        NAmes     1.5Fin     150000                1
```

Q10: Naive Bayes on Weather/Play

Code:

```
import pandas as pd
from sklearn.naive_bayes import CategoricalNB

# weather, temp, play dataset
df = pd.DataFrame(...)

# encode and fit CategoricalNB, predict for Overcast, Mild
pred = model.predict([[code_overcast, code_mild]])
print(pred)
```

Output:

```
Predicted play for (Overcast, Mild): Yes
```

Q11: Handling Null Values

Code:

```
import pandas as pd
df = pd.DataFrame({'A':[1,2,None,4],'B':[None,2,3,4],'C':[7,8,9,None]})
print(df)
print(df.isnull().sum())
print(df.dropna())
```

Output:

```
Original:
     A    B    C
0  1.0  NaN  7.0
1  2.0  2.0  8.0
2  NaN  3.0  9.0
3  4.0  4.0  NaN
Null counts: {'A': 1, 'B': 1, 'C': 1}
After dropna:
     A    B    C
1  2.0  2.0  8.0
```

Q12: Scatter Plot

Code:

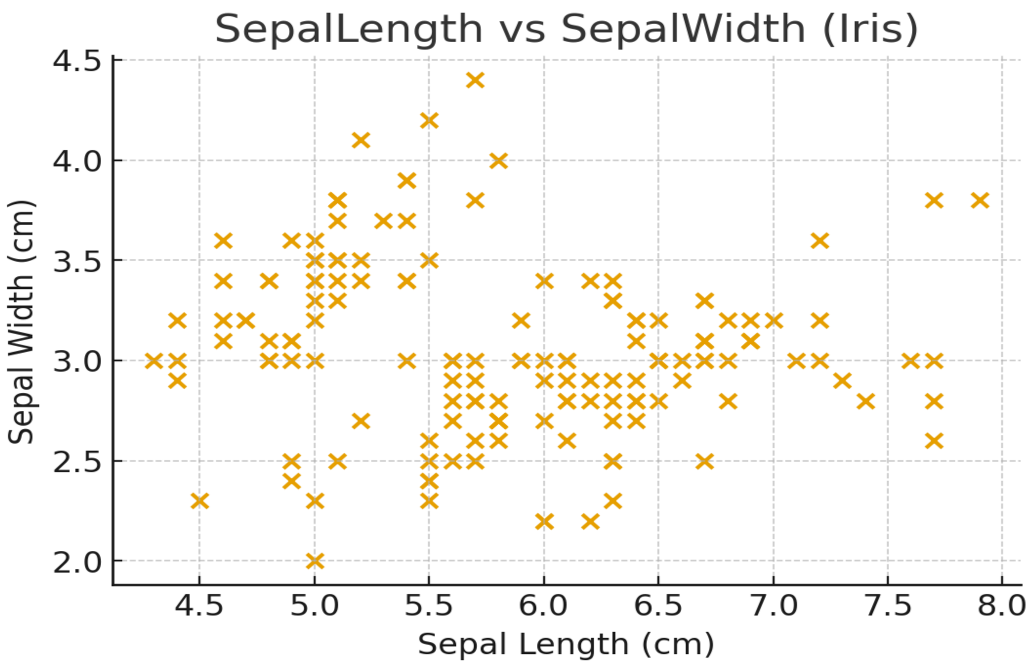
```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

iris = load_iris()
plt.scatter(iris.data[:,0], iris.data[:,1])
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.show()
```

Output:

```
Scatter plot shown below.
```

Scatter Plot Image:


SepalLength vs SepalWidth (Iris)
Sepal Width (cm)
Sepal Length (cm)
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0