Machine Learning (Summer 2019)
Midterm Exam

الاسم: ..... رقم التسجيل: .....

==========

Instructions: Time **60** min. Open book and notes exam. No electronics. Please answer all problems in the space provided and limit your answer to the space provided. No questions are allowed. There are five problems.

==========

P1. Give five machine learning solutions that are used in our everyday life. Also, for every solution, specify the type of used machine learning according to human supervision criterion.

[5 points]

1) **Google search, supervised**

2) **Google translate, supervised**

3) **Google photo people tagging, semi-supervised**

4) **Voice command, supervised**

5) **Amazon book suggestion, unsupervised**

P2. Write a Python function that accepts a list of string benefits and prints each benefit on one row. This function should call a sub-function named `build_sentence(n, benefit)` that receives an integer *n* and a string *benefit* and returns a sentence "Job Benefit *n*: *benefit*". For example, when the function is called with `print_the_benefits(["JOD600 Salary", "Health Insurance"])`, it should print:

```
Job Benefit 1: JOD600 Salary
Job Benefit 2: Health Insurance
```

[5 points]

```
# your code goes here

def build_sentence(n, benefit):
    return "Job Benefit %d: %s" % (n, benefit)

def print_the_benefits (list_of_benefits):
    for n, benefit in enumerate(list_of_benefits):
     print(build_sentence(n+1, benefit))
```

P3. Write Python code to convert temperature list from Python list to a Numpy array. Then, convert all of the temperature values from Fahrenheit to Celsius. Use the formula $\boldsymbol{C = (F - 32) \times 5/9}$ to make your conversion. Lastly, print the resulting array of temperature values in Celsius.

[5 points]

```
temp_F = [81.65, 97.52, 95.25, 92.98, 86.18, 88.45]

# your code goes here

import numpy as np

# Create a numpy array np_temp_F from temp_F
np_temp_F = np.array(temp_F)

# Create np_temp_C from np_temp_F
np_temp_C = (np_temp_F - 32.) * 5. / 9.

# Print out np_temp_C
print(np_temp_C)
```

P4. Given the information in the box blow, complete the following Python code to scale the numerical feature, handle the categorical feature, train an SVM Regressor using the prepared features, predict the response `y` from the prepared data, and print the RMSE of the predicted response.

[10 points]

```
>>> data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 3 columns):
x1          30000 non-null float64
x2          30000 non-null object
y           30000 non-null float64
dtypes: float64(3)
memory usage: 1.8+MB
>>> data["x2"].value_counts()
Bird    10000
Cat     10000
Dog     10000
Name: x2, dtype: int64
```

```
from sklearn.preprocessing import StandardScaler, LabelBinarizer
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error

X = data.drop("y", axis=1)
y = data["y"].copy()

# your code goes here

Import numpy as np

X_num = X.drop("x2", axis=1)
X_cat = X.drop("x1", axis=1)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_num)

encoder = LabelBinarizer()
X_cat_1hot = encoder.fit_transform(X_cat)

X_prepared = np.c_[X_scaled, X_cat_1hot]

svm_reg = SVR()
svm_reg.fit(X_prepared, y)

y_predictions = svm_reg.predict(X_prepared)

print(np.sqrt(mean_squared_error(y, y_predictions)))
```

P5. Complete the following Python code to train a binary classifier that detects digits smaller than 5 using the Stochastic Gradient Descent (SGD) classifier. Train this classifier on the training set, predict the classes of the test set, find the confusion matrix, calculate the prediction accuracy from the confusion matrix, and print this accuracy.

[5 points]

```
from sklearn.datasets import fetch_mldata
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import confusion_matrix

mnist = fetch_mldata('MNIST original')
X, y = mnist["data"], mnist["target"]
X_train, X_test = X[:60000], X[60000:]
y_train, y_test = y[:60000], y[60000:]

# your code goes here

y_train_less_5 = (y_train < 5)
y_test_less_5 = (y_test < 5)

sgd_clf = SGDClassifier()
sgd_clf.fit(X_train, y_train_less_5)

y_pred = sgd_clf.predict(X_test)

cm = confusion_matrix(y_test_less_5, y_pred)

acc = (cm[0][0]+cm[1][1]) / sum(sum(cm)) #10000
print('Accuracy = ', acc)
```

<Good Luck>