Magnimind-Fullstack **Name: ARIF AYGUN**
Applied Machine Learning

Time Limit:

1. This exam contains 7 pages (including this cover page) and 22 questions.
2. Total of points is 100.
3. Answer all questions. The marks for each question are indicated at the beginning of each question.
4. This **IS an OPEN BOOK** exam.
5. Calculators are not allowed.

1 True/False (20 points)

1. (2 points) In L1 regularization, we penalize the absolute value of the weights while in L2 regularization, we penalize the squared value of the weights. TRUE
2. (2 points) The scikit-learn Python machine learning library provides the ColumnTransformer that allows you to selectively apply data transforms to different columns in your dataset TRUE
3. (2 points) It is good practice to use MinMaxScaling on a feature with a few extreme Outliers. FALSE
4. (2 points) The parameter k should take odd values in kNN, so that there are no ties in the voting. TRUE
5. (2 points) Treating a non-ordinal categorical variable as continuous variable would result in a better predictive model. FALSE
6. (2 points) One-hot-encoding increasing the dimensionality of a data set. TRUE
7. (2 points) OLS Regression is expected to have more overfitting (lower bias) than Ridge. FALSE
8. (2 points) Fitting your scaling transformation separately to your training and the test sets improves the model performance. FALSE
9. (2 points) The more features that we use to represent our data, the better the learning algorithm will generalize to new data points. FALSE
10. (2 points) It is not a good machine learning practice to use the test set to help adjust the hyperparameters of your learning algorithm TRUE

2 Multiple Choice (20 points, 4pts each)

Select **all** choices that apply.

11. (4 points) Cross validation:

    A. Does nothing to prevent overfitting

    B. **Is often used to select hyperparameters**

    C. Is guaranteed to prevent overfitting

    D. None of the above

12. (4 points) Say, there are two kids Jack and Jill in a maths exam. Jack only learnt additions and Jill memorized the questions and their answers from the maths book. Now, who will succeed in the exam? The answer is neither. From machine learning lingo, Jack is **blank1** and Jill is **blank2**.

    A. blank1=overfitting, blank2=underfitting

    B. **blank1=underfitting, blank2=overfitting**

    C. blank1=underfitting, blank2=underfitting

    D. **blank1=overfitting, blank2=overfitting**

13. (4 points) You are working on a *classification problem*. For validation purposes, you have randomly sampled the training data set into train and validation. You are confident that your model will work incredibly well on unseen data since your validation accuracy is high. However, you get shocked after getting poor test accuracy. What might have gone wrong?

    A. You trained an OLS model.

    B. You are using an imbalanced dataset.

    C. You used the simplest model you could come up with

    D. **You used random sampling instead of stratified sampling when forming the test and train datasets.**

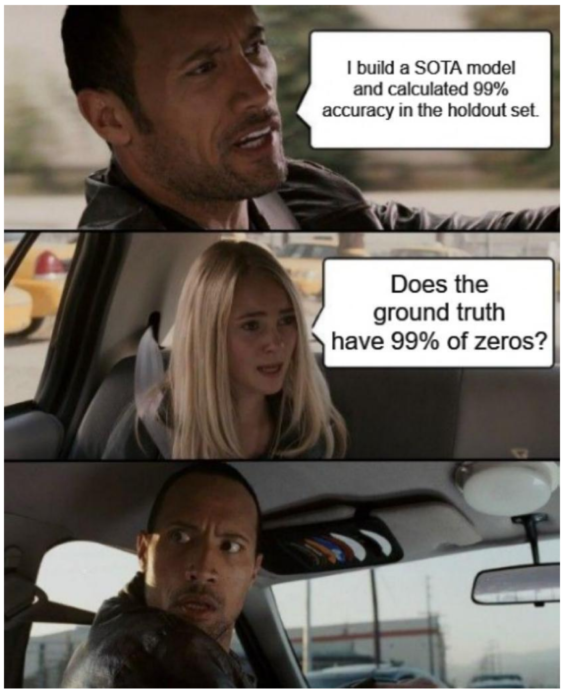
14. (4 points) Which of the following models is more acceptable then others to be implemented on new data points ?

    A. My model achieves a training error lower than all previous models!

    B. **My modelachieves a test error lower than all previous models! (When regularization parameter α is chosen so as to minimize test error.)**

    C. **My method achieves a cross-validation error lower than all previous methods! (When regularization parameter α is chosen so as to minimize crossvalidaton error.)**

    D. My method achieves a test error lower than all previous methods! (When regularization parameter α is chosen so as to minimize cross-validaton error.)


I build a SOTA model and calculated 99% accuracy in the holdout set.
Does the ground truth have 99% of zeros?

15. (4 points) Which of the following explains the worrisome face of the driver at the end? (Note: Holdout set=testing data)

    A. He is about to run out of gas.

    B. His State of the Art (SOTA) model was not designed for sparse datasets.

    C. **He did not use his holdout set when training the model, so he has no idea how the model gave him a good result, and he is now really worried that he looks silly in front of his girlfriend.**

    D. **His dataset may be imbalanced, he didn't check it, so his accuracy on the model may not mean much about the performance of the model on the holdout set.**

3 Debugging

For each code snippet, find and explain **all** errors given the task.Assume all necessary classes are imported.There can be more than one errors.You can write your answer on the empty spaces on this page.

16. (10 points) Task: Perform cross validation to pick the best model for LinearRegression() using cv=10 and then print the accuracy of the model on the test data set.

```
 X_train, X_test, y_train, y_test=train_test_split(X,y)
mycross= cross_val_score(LinearRegression(), X_train, y_train, cv=10)
print("The accuracy is: {}".format(mycross.score(X_test,y_test))
```

17. (10 points) Task: Create a pipeline to preprocesss data by scaling and adding polynomial features and then learn a Ridge model with GridSearchCV .

```
X_train, X_test, y_train, y_test = train_test_split( X,y, random_state=0)
pipe = make_pipeline(StandardScaler(),PolynomialFeatures(), Ridge())
param_grid = {‘mypolynomial__degree’: [1, 2, 3],
                    ‘myridge__alpha’: [0.001, 0.01, 0.1, 1, 10, 100]}
grid = GridSearchCV(pipe, param_grid=param_grid, return_train_score=True)
Ridge.fit(X_train, y_train)
```

16.

1. Error in train_test_split():
The train_test_split() function is missing the argument for the test_size. This function splits the data into training and testing sets, so it requires specifying the proportion or number of samples to be allocated to the test set. The corrected line:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2) # Adjust the test_size as needed

2. Error in cross_val_score():
The cross_val_score() function is used to perform cross-validation and evaluate the model's performance. However, the code snippet is trying to access the "score" method of the cross_val_score result, which does not exist. Instead, the cross_val_score() returns an array of scores for each fold. To fix this error, we can simply print the mean of the cross-validation scores. The corrected lines:

mycross = cross_val_score(LinearRegression(), X_train, y_train, cv=10)
print("The mean accuracy is: {}".format(mycross.mean()))

3. Error in accessing the score on test data:
The LinearRegression() model does not have a "score" method to calculate accuracy. The "score" method is typically used for classification tasks, not regression tasks. In regression tasks, metrics like Mean Squared Error (MSE) or R-squared are commonly used to evaluate model performance. To calculate the model's performance on the test data, we can use the model's predict() method to generate predictions and then calculate an appropriate regression metric. The corrected lines:

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error on test data: {}".format(mse))

17.

1. Error in assigning train-test split:
The train_test_split() function is missing the test_size argument. This function splits the data into training and testing sets, so it requires specifying the proportion or number of samples to be allocated to the test set.

The corrected line:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

2. Error in pipeline creation:
The make_pipeline() function is used to create a pipeline with multiple steps. However, the steps in the pipeline are not assigned any names or aliases. To fix this error, you should assign names to the steps using the format <step_name>__<parameter_name>. In this case, we can use "scaler" as the name for StandardScaler, "poly" for PolynomialFeatures, and "ridge" for Ridge.

The corrected line:

pipe = make_pipeline(StandardScaler(), PolynomialFeatures(), Ridge())

3. Error in defining the parameter grid:
The parameter grid provided for GridSearchCV contains incorrect syntax for specifying the parameters of PolynomialFeatures() and Ridge(). Instead of mypolynomial__degree and myridge__alpha, the correct syntax is polynomialfeatures__degree and ridge__alpha.

The corrected param_grid:

param_grid = {'polynomialfeatures__degree': [1, 2, 3], 'ridge__alpha': [0.001, 0.01, 0.1, 1, 10, 100]}

4. Error in fitting the Ridge model:
The code snippet is calling the fit() method on the Ridge class directly, instead of using the fitted pipeline (pipe).

The corrected line:

pipe.fit(X_train, y_train)

the code will perform the desired task of creating a pipeline with scaling, polynomial feature generation, and Ridge model fitting using GridSearchCV.

4 Coding

Assume all necessary imports are already made.

18. (10 points) Provide code to evaluate Linear Regression (OLS), Ridge, and Lasso using cross-validation with the default parameters on a separate test set, given a dataset as numpy arrays X and y.

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import train_test_split

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Create Linear Regression (OLS) model and evaluate using cross-validation
ols = LinearRegression()
ols_scores = cross_val_score(ols, X_train, y_train, cv=5)

# Create Ridge model and evaluate using cross-validation
ridge = Ridge()
ridge_scores = cross_val_score(ridge, X_train, y_train, cv=5)

# Create Lasso model and evaluate using cross-validation
lasso = Lasso()
lasso_scores = cross_val_score(lasso, X_train, y_train, cv=5)

# Print the cross-validation scores
print("Linear Regression (OLS) CV scores:", ols_scores)
print("Ridge CV scores:", ridge_scores)
print("Lasso CV scores:", lasso_scores)

# Fit the models on the entire training set
ols.fit(X_train, y_train)
ridge.fit(X_train, y_train)
lasso.fit(X_train, y_train)

# Evaluate the models on the test set
ols_test_score = ols.score(X_test, y_test)
ridge_test_score = ridge.score(X_test, y_test)
lasso_test_score = lasso.score(X_test, y_test)

# Print the test set scores
print("Linear Regression (OLS) test score:", ols_test_score)
print("Ridge test score:", ridge_test_score)
print("Lasso test score:", lasso_test_score)
```

19. (10 points) Tune the parameters of the models above using GridSearchCV.Visualize the dependence of the validation score on the parameters for Ridge and Lasso.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge, Lasso

# Define the parameter grid for Ridge
ridge_param_grid = {'alpha': np.logspace(-3, 3, 7)}

# Create Ridge model
ridge = Ridge()

# Create GridSearchCV object for Ridge with parameter grid
ridge_grid = GridSearchCV(ridge, param_grid=ridge_param_grid, cv=5)
ridge_grid.fit(X, y)

# Get the results of GridSearchCV for Ridge
ridge_results = ridge_grid.cv_results_
ridge_params = ridge_results['params']
ridge_mean_scores = ridge_results['mean_test_score']

# Plot the validation scores for Ridge
plt.figure()
plt.semilogx(ridge_params, ridge_mean_scores)
plt.xlabel('Alpha')
plt.ylabel('Validation Score')
plt.title('Validation Scores for Ridge')
plt.grid(True)
plt.show()

# Define the parameter grid for Lasso
lasso_param_grid = {'alpha': np.logspace(-3, 3, 7)}

# Create Lasso model
lasso = Lasso()

# Create GridSearchCV object for Lasso with parameter grid
lasso_grid = GridSearchCV(lasso, param_grid=lasso_param_grid, cv=5)
lasso_grid.fit(X, y)

# Get the results of GridSearchCV for Lasso
lasso_results = lasso_grid.cv_results_
lasso_params = lasso_results['params']
lasso_mean_scores = lasso_results['mean_test_score']

# Plot the validation scores for Lasso
plt.figure()
plt.semilogx(lasso_params, lasso_mean_scores)
plt.xlabel('Alpha')
plt.ylabel('Validation Score')
plt.title('Validation Scores for Lasso')
plt.grid(True)
plt.show()
```

5 Concepts (20 Points)

20. (5 points) What is the difference between supervised and unsupervised machine learning?

The main difference between supervised and unsupervised machine learning lies in the presence or absence of labeled data during the training process:

Supervised Learning:

1. Supervised learning involves training a model using labeled data, where the input features (X) are associated with corresponding target variables or labels (y).

2. The goal of supervised learning is to learn a mapping function from input features to output labels.

3. The model is trained on a labeled dataset, where both the input features and the correct labels are provided.

4. During training, the model learns patterns and relationships in the data and tries to generalize to make accurate predictions on unseen data.

5. Examples of supervised learning algorithms include linear regression, logistic regression, decision trees, support vector machines (SVM), and neural networks.

Unsupervised Learning:

1. Unsupervised learning involves training a model on unlabeled data, where only the input features (X) are available without any corresponding labels (y).

2. The goal of unsupervised learning is to discover patterns, relationships, and structures in the data without prior knowledge of the correct labels.

3. The model learns from the inherent structure of the data to identify clusters, anomalies, or hidden patterns.

4. Unsupervised learning is often used for exploratory data analysis, dimensionality reduction, and feature extraction.

5. Examples of unsupervised learning algorithms include clustering algorithms like k-means, hierarchical clustering, and DBSCAN, as well as dimensionality reduction techniques like principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE).

In summary, supervised learning relies on labeled data to learn patterns and make predictions, while unsupervised learning focuses on extracting information from unlabeled data and discovering hidden patterns or structures.

21. (5 points) When should you use classification over regression?

Classification and regression are two fundamental tasks in machine learning that serve different purposes and are applied in different scenarios. Here is a detailed explanation of when to use classification over regression:

Nature of the Target Variable:

Classification: Use classification when the target variable or the outcome of interest is categorical in nature. In classification, the goal is to predict the class or category to which an observation belongs.

Regression: Use regression when the target variable is continuous or numeric. In regression, the goal is to predict a numerical value or estimate a relationship between variables.

Type of Problem:

Classification: Use classification when you want to solve problems like image classification, email spam detection, sentiment analysis, fraud detection, disease diagnosis, or any other task where you need to classify data into different categories or classes.

Regression: Use regression when you want to solve problems like predicting housing prices, stock market analysis, demand forecasting, or any other task where the goal is to estimate a numerical value or make continuous predictions.

Data Representation:

Classification: Classification algorithms are designed to handle categorical features as inputs and predict the class labels. These algorithms are trained to learn the decision boundaries that separate different classes in the feature space.
Regression: Regression algorithms work with numerical features as inputs and aim to find the best-fitting line or curve that represents the relationship between the input features and the target variable.

Evaluation Metrics:

Classification: Classification models are evaluated using metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC). These metrics measure the model's ability to correctly classify instances into their respective classes.
Regression: Regression models are evaluated using metrics such as mean squared error (MSE), mean absolute error (MAE), root mean squared error (RMSE), and R-squared. These metrics quantify the model's ability to predict continuous values and measure the goodness of fit between the predicted and actual values.

Decision Making:

Classification: Classification models provide a clear decision boundary and can help in making binary or multi-class decisions. For example, in medical diagnosis, a classification model can determine whether a patient has a particular disease or not.
Regression: Regression models provide a continuous predicted value and can help in understanding the relationship between variables. For example, in sales forecasting, a regression model can estimate the expected sales volume based on various factors.

In summary, use classification when dealing with categorical target variables, solving classification problems, working with categorical features, and evaluating the model's classification performance. Use regression when dealing with continuous target variables, solving regression problems, working with numerical features, and evaluating the model's prediction accuracy on continuous values.

22. (10 points) In dermatology, correctly diagnosing various types of skin conditions can be challenging, since the symptoms of the different conditions can be quite similar. The skin conditions that we consider are psoriasis, seboreic dermatitis, lichen planus, pityriasis rosea, cronic dermatitis, and pityriasis rubra pilaris. Determining the exact condition can sometimes require a biopsy, which can be expensive. In order to speed up the process of making a diagnosis, we'd like to develop an automatic classifier. For a large number of patients, we describe their symptoms using 33 attributes. These attributes are a mix of categorical attributes (e.g. the patient's gender, whether or not the skin itches) and numerical attributes (e.g. the patient's age, the size of the affected part). For each of these patients, a diagnosis has been determined: one of the six conditions mentioned above, or a special label Other for other types of diagnoses. Explain how you would implement a machine learning model that would solve this prediction task. You don't need to show Python code, but please give a description of the system and explain all steps you would carry out when developing it.

To develop a machine learning model for the diagnosis of skin conditions based on the given symptoms, the following steps can be taken:

Data Collection and Preparation:
1.Gather a large dataset of patients' information, including the 33 attributes describing their symptoms and the corresponding diagnosis.
2.Ensure the dataset is representative and diverse, covering various age groups, genders, and skin conditions.
3.Preprocess the data by handling missing values, encoding categorical variables, and normalizing numerical attributes if required.
Exploratory Data Analysis:
1.Perform exploratory data analysis to understand the distribution of the attributes, identify any patterns or correlations, and gain insights into the data.
2.Visualize the data to observe any discernible patterns or relationships between attributes and the diagnosis.
Feature Selection and Engineering:
1.Select relevant features that are likely to contribute to the prediction task based on domain knowledge and exploratory analysis.
2.Perform feature engineering if necessary, such as creating new features, transforming variables, or applying dimensionality reduction techniques to improve model performance.
Model Selection:
1.Choose an appropriate machine learning algorithm that is suitable for multi-class classification tasks and can handle a mix of categorical and numerical attributes.
2.Common algorithms that can be considered include decision trees, random forests, support vector machines (SVM), or gradient boosting algorithms.
Model Training and Evaluation:
1.Split the dataset into training and testing sets, ensuring an appropriate ratio to have enough data for model training and evaluation.
2.Train the selected machine learning model on the training set using the symptoms attributes as input and the corresponding diagnosis labels as the target variable.
3.Evaluate the model's performance on the testing set using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.
4.Use cross-validation techniques to assess the model's generalization capability and to tune hyperparameters if needed.
Model Optimization:
1.Fine-tune the model's hyperparameters using techniques like grid search or randomized search to improve its performance.
2.Consider using techniques like oversampling or undersampling if there is an imbalance in the distribution of different skin conditions in the dataset.
Deployment and Monitoring:
1.Once the model achieves satisfactory performance, deploy it in a production environment to make predictions on new, unseen data.
2.Continuously monitor the model's performance and retrain it periodically with new data to ensure its accuracy and reliability.

It is important to note that developing an accurate machine learning model for medical diagnosis requires collaboration with dermatologists or medical professionals to ensure the model's reliability, interpretability, and adherence to ethical guidelines.