**Составление тестовых заданий**

**ML 0.1**

**№1**

**Изучите данный фрагмент кода:**

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

**Предположим, у вас есть датасет df с признаками X и метками y.**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Обучение модели случайного леса**

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

**Метод, который используется для оценки важности признаков в модели RandomForestClassifier, называется:**

a) model.coef\_

b) model.feature\_importances\_ %100%

c) model.weights\_

d) model.importance\_

e) model.attribute\_importance\_

**№2**

**Изучите данный фрагмент кода:**

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import Pipeline

from sklearn.model\_selection import cross\_val\_score

**Предположим, у вас есть датасет df с признаками X и метками y.**

**Вы хотите применить логистическую регрессию с кросс-валидацией.**

**Для построения пайплайна с использованием StandardScaler и LogisticRegression используятся шаги:**

a) pipeline = Pipeline([('scaler', StandardScaler()), ('model', LogisticRegression())]) %50%

b) pipeline = Pipeline([('model', LogisticRegression()), ('scaler', StandardScaler())])

c) pipeline = Pipeline([('scaler', StandardScaler()), ('classifier', LogisticRegression())]) %50%

d) pipeline = Pipeline([('classifier', LogisticRegression()), ('scaler', StandardScaler())])

e) pipeline = Pipeline([('model', StandardScaler()), ('classifier', LogisticRegression()), ('scaler', StandardScaler())])

f) pipeline = Pipeline([('scaler', LogisticRegression()), ('model', StandardScaler())])

**ML 0.2**

**№1**

**Дана строка кода:**

import pandas as pd

**Предположим, у вас есть DataFrame df с данными о продажах товаров.**

**Метод, который используется для получения основных статистических характеристик (среднее, стандартное отклонение, и т. д.) числовых столбцов, называется:**

a) df.describe() %100%

b) df.info()

c) df.head()

d) df.mean()

e) df.stats()

**№2**

**Ознакомьтесь с фрагментом кода:**

import seaborn as sns

import matplotlib.pyplot as plt

**Предположим, у вас есть DataFrame df с данными о росте и весе людей.**

**Методы, которые могут быть использованы для визуализации корреляции между ростом и весом:**

a) sns.scatterplot(x='height', y='weight', data=df) %33%

b) sns.heatmap(df.corr(), annot=True) %33%

c) sns.barplot(x='height', y='weight', data=df)

d) plt.plot(df['height'], df['weight'])

e) sns.pairplot(df[['height', 'weight']]) %33%

f) sns.lineplot(x='height', y='weight', data=df)

**ML 1.1**

**№1**

**Изучите данный фрагмент кода:**

import pandas as pd

from sklearn.linear\_model import LinearRegression

**Предположим, у вас есть DataFrame df с данными о времени подготовки к экзамену (X - часы) и оценке за экзамен (y).**

**Метод для обучения парной линейной регрессии с использованием библиотеки scikit-learn – это:**

a) model = LinearRegression.fit(X, y)

b) model = LinearRegression.predict(X, y)

c) model = LinearRegression.train(X, y)

d) model = LinearRegression() %100%

model.fit(X, y)

**№2**

**Дан фрагмент кода:**

import seaborn as sns

import matplotlib.pyplot as plt

**Предположим, у вас есть DataFrame df с данными о времени подготовки к экзамену (X - часы) и оценке за экзамен (y).**

**Методы, которые могут быть использованы для визуализации парной линейной регрессии:**

a) sns.scatterplot(x='hours\_study', y='exam\_score', data=df)

b) sns.lmplot(x='hours\_study', y='exam\_score', data=df) %50%

c) sns.lineplot(x='hours\_study', y='exam\_score', data=df)

d) plt.plot(df['hours\_study'], df['exam\_score'])

e) sns.regplot(x='hours\_study', y='exam\_score', data=df, order=1) %50%

f) sns.barplot(x='hours\_study', y='exam\_score', data=df)

**ML 1.2**

**№1**

**Дан фрагмент кода:**

import pandas as pd

from sklearn.linear\_model import LinearRegression

**Предположим, у вас есть DataFrame df с данными о времени подготовки к экзамену (X1 - часы) и количестве учебных материалов (X2 - шт.), а также оценке за экзамен (y).**

**Метод, который используется для обучения множественной линейной регрессии с использованием библиотеки scikit-learn – это:**

a) model = LinearRegression.fit(X1, X2, y)

b) model = LinearRegression.predict(X1, X2, y)

c) model = LinearRegression.train(X1, X2, y)

d) model = LinearRegression() %100%

model.fit(X, y)

**№2**

**Дан фрагмент кода:**

import seaborn as sns

import matplotlib.pyplot as plt

**Предположим, у вас есть DataFrame df с данными о времени подготовки к экзамену (X1 - часы) и количестве учебных материалов (X2 - шт.), а также оценке за экзамен (y).**

**Методы, которые могут быть использованы для визуализации множественной линейной регрессии:**

a) sns.scatterplot(x='hours\_study', y='materials', hue='exam\_score', data=df)

b) sns.lmplot(x='hours\_study', y='exam\_score', hue='materials', data=df) %50%

c) sns.lineplot(x='hours\_study', y='exam\_score', data=df)

d) sns.pairplot(df[['hours\_study', 'materials', 'exam\_score']]) %50%

e) sns.regplot(x='hours\_study', y='exam\_score', data=df, order=2)

f) sns.barplot(x='hours\_study', y='exam\_score', hue='materials', data=df)

**ML 1.3**

**№1**

**Дан фрагмент кода:**

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import Ridge

**Предположим, у вас есть датасет df с признаками X и метками y.**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Метод, который используется для обучения модели регрессии с использованием регуляризации L2 (Ridge regression) в scikit-learn – это:**

a) model = Ridge.fit(X\_train, y\_train)

b) model = Ridge.predict(X\_train, y\_train)

c) model = Ridge.train(X\_train, y\_train)

d) model = Ridge() %100%

model.fit(X\_train, y\_train)

**№2**

**Дан фрагмент кода:**

import seaborn as sns

import matplotlib.pyplot as plt

**Предположим, у вас есть DataFrame df с данными о времени подготовки к экзамену (X1 - часы) и количестве учебных материалов (X2 - шт.), а также оценке за экзамен (y).**

**Следующие модели регрессии могут быть применены к данному датасету:**

a) Линейная регрессия %20%

b) Регрессия с использованием полиномиальных признаков %20%

c) Логистическая регрессия

d) Ridge регрессия %20%

e) DecisionTreeRegressor %20%

f) KNeighborsRegressor %20%

**ML 2.1**

**№1**

**Дан фрагмент кода:**

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

**Предположим, у вас есть датасет df с признаками X и бинарной меткой класса y (0 или 1).**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Метод, который используется для обучения модели логистической регрессии в scikit-learn.**

a) model = LogisticRegression.fit(X\_train, y\_train)

b) model = LogisticRegression.predict(X\_train, y\_train)

c) model = LogisticRegression.train(X\_train, y\_train)

d) model = LogisticRegression() %100%

model.fit(X\_train, y\_train)

**№2**

**Дан фрагмент кода:**

import seaborn as sns

import matplotlib.pyplot as plt

**Предположим, у вас есть DataFrame df с данными о времени подготовки к экзамену (X1 - часы) и результате сдачи экзамена (y - сдал/не сдал).**

**Верные утверждения относительно модели логистической регрессии:**

a) Логистическая регрессия используется для задачи бинарной классификации. %25%

b) Логистическая регрессия выдаёт вероятности принадлежности к каждому классу. %25%

c) Основной функцией потерь для логистической регрессии является Mean Squared Error.

d) Логистическая регрессия может быть расширена для многоклассовой классификации. %25%

e) Веса модели логистической регрессии могут быть интерпретированы как вклад каждого признака в предсказание. %25%

f) Логистическая регрессия подходит только для задач регрессии, а не классификации.

**ML 2.2**

**№1**

**Дан фрагмент кода:**

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

**Предположим, у вас есть датасет df с признаками X и бинарной меткой класса y (0 или 1).**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Алгоритм, использующийся для обучения модели классификации на основе ансамбля деревьев решений в scikit-learn – это:**

a) DecisionTreeClassifier

b) Support Vector Machine (SVM)

c) K-Nearest Neighbors (KNN)

d) Random Forest Classifier %100%

e) Logistic Regression

**№2**

**Дан фрагмент кода:**

import seaborn as sns

import matplotlib.pyplot as plt

**Предположим, у вас есть DataFrame df с данными о результатах экзамена (X1 - оценка) и статусе поступления в университет (y - поступил/не поступил).**

**Верные из следующих утверждений относительно моделей классификации:**

a) Модель классификации может быть использована для прогнозирования вероятности отнесения к каждому классу. %25%

b) ROC-кривая (Receiver Operating Characteristic) может быть использована для оценки качества модели классификации. %25%

c) Метод k-Nearest Neighbors (KNN) является линейной моделью классификации.

d) F1-score - это комбинированная метрика, учитывающая как точность, так и полноту модели. %25%

e) В задаче бинарной классификации Precision (точность) рассчитывается как отношение правильно предсказанных положительных к общему числу положительных случаев. %25%

f) В Random Forest Classifier каждое дерево строится на основе всего набора данных.