Data Mining

Team 2

Chong Shen, Peifeng Li, Jiajun Chen, Xiangnan Zhang, Kebin Li, Fan Shen

THIS HOMEWORK IS DONE IN MySQL

```
USE Datasets;

CREATE TABLE Challenger
(o_ring_failure INT,
launch_temperature INT,
leak_check_pressure VARCHAR(10)
);

LOAD DATA LOCAL INFILE '/Users/bryton/Desktop/Challenger.csv' INTO TABLE
Challenger
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;

SELECT * FROM Challenger;
```

o_ring_failure	launch_temperature	leak_check_pressure
0	66	Low
1	70	Low
0	69	Low
0	68	Low
0	67	Low
0	72	Low
0	73	Medium
0	70	Medium
1	57	High
1	63	High
1	70	High
0	78	High
0	67	High
1	53	High
0	67	High
0	75	High
0	70	High
0	81	High
0	76	High
0	79	High
0	75	High
0	76	High
1	58	High

1. Build a univariate user defined function for each of the following univariate statistics:

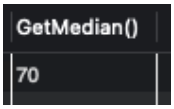
Range

```
DELIMITER $$
CREATE FUNCTION GetRange()
RETURNS VARCHAR(30)
BEGIN
DECLARE v INT;
DECLARE n INT;
SELECT MIN(launch_temperature), MAX(launch_temperature) FROM Challenger INTO v, n;
RETURN CONCAT(v, '-', n);
END$$

SELECT GetRange();
```

GetRange()
53-81

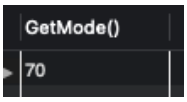
Median

```
DELIMITER $$
CREATE FUNCTION GetMedian()
RETURNS FLOAT
BEGIN
SET @row_index := -1;
RETURN
(SELECT AVG(subq.launch_temperature) AS median
FROM (SELECT @row_index:= @row_index +1 AS row_index, launch_temperature
FROM Challenger
ORDER BY launch_temperature
            ) AS subq
        WHERE subq.row_index
        IN (FLOOR(@row_index / 2 ), CEIL(@row_index/2)));
END$$

SELECT GetMedian();
```



Mode

```
DELIMITER $$
CREATE FUNCTION GetMode()
RETURNS VARCHAR(10)
BEGIN
RETURN (SELECT launch_temperature as M
FROM Challenger
GROUP BY launch_temperature
ORDER BY COUNT(launch_temperature) DESC
LIMIT 1);
END$$

SELECT GetMode();
```

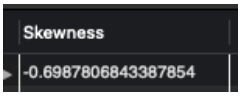


The mode of launch temperature is 70, and the value of mode is 4.

Skewness

```
CREATE FUNCTION GetSkewness()
RETURNS FLOAT
BEGIN
DECLARE res FLOAT;
WITH SkewCTE AS
(
SELECT SUM(1.0*launch_temperature) AS rx,
SUM(POWER(1.0*launch_temperature,2)) AS rx2,
SUM(POWER(1.0*launch_temperature,3)) AS rx3,
COUNT(1.0*launch_temperature) AS rn,
STDDEV(1.0*launch_temperature) AS stdv,
AVG(1.0*launch_temperature) AS av
FROM Challenger
)
SELECT
```

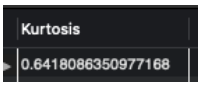
```
   (rx3 - 3*rx2*av + 3*rx*av*av - rn*av*av*av)
   / (stdv*stdv*stdv) * rn / (rn-1) / (rn-2) AS Skewness
FROM SkewCTE INTO res;
RETURN res;
END
```



The skewness value is -0.69 in the range -0.5 and -1, thus the distribution of launch temperature is moderately skewed

Kurtosis

```
CREATE FUNCTION GetKurtosis()
RETURNS FLOAT
BEGIN
DECLARE res FLOAT;
WITH KurtCTE AS
(
SELECT SUM(1.0*launch_temperature) AS rx,
SUM(POWER(1.0*launch_temperature,2)) AS rx2,
SUM(POWER(1.0*launch_temperature,3)) AS rx3,
SUM(POWER(1.0*launch_temperature,4)) AS rx4,
COUNT(1.0*launch_temperature) AS rn,
STDDEV(1.0*launch_temperature) AS stdv,
AVG(1.0*launch_temperature) AS av
FROM Challenger
)
SELECT
   (rx4 - 4*rx3*av + 6*rx2*av*av - 4*rx*av*av*av + rn*av*av*av*av)
   / (stdv*stdv*stdv*stdv) * rn * (rn+1) / (rn-1) / (rn-2) / (rn-3)
   - 3.0 * (rn-1) * (rn-1) / (rn-2) / (rn-3) AS Kurtosis
FROM KurtCTE INTO res;
RETURN res;
END
```



Notice that we already -3 in our kurtosis formula, we get a positive Kurtosis value of 0.642. Thus, the distribution of launch temperature is leptokurtic. Compare to a normal distribution, its tails are longer and fatter, and often its central peak is higher and sharper.

2. Build a bivariate stored procedure for Z Test:

Input: Table name, Categorical Name, Numerical Column Name
Output: Z statistics and P(Z)

*****Note*****
Please note that MySQL do not support passing the table name in a procedure
So we do not set any parameters in our procedure

```
CREATE PROCEDURE ZScore()
BEGIN
DECLARE res FLOAT;
```

```
DECLARE av1 FLOAT;
DECLARE av0 FLOAT;
DECLARE d1 FLOAT;
DECLARE d0 FLOAT;
DECLARE num1 FLOAT;
DECLARE num0 FLOAT;
SELECT AVG(1.0*launch_temperature) FROM Challenger WHERE o_ring_failure=1 INTO
av1;
SELECT AVG(1.0*launch_temperature) FROM Challenger WHERE o_ring_failure=0 INTO
av0;
SELECT POWER(STDDEV(1.0*launch_temperature),2) AS stdv FROM Challenger WHERE
o_ring_failure=1 INTO d1;
SELECT POWER(STDDEV(1.0*launch_temperature),2) AS stdv FROM Challenger WHERE
o_ring_failure=0 INto d0;
SELECT COUNT(launch_temperature) FROM Challenger WHERE o_ring_failure=1 INTO num1;
SELECT COUNT(launch_temperature) FROM Challenger WHERE o_ring_failure=0 INTO num0;
SELECT (av0-av1)/POWER((d1/num1+d0/num0),0.5) AS z;
END

CALL ZScore();
```

z
3.650476019206606

According to the Z table, we can know P(z) = 0.00013.

3. Build a stored procedure for K Nearest Neighbors

- **Input: (Temperature, K)**
- **Output: K nearest neighbors (Target) with the corresponding probability**
- **The space shuttle Challenger disaster happened when temperature was 30 before launch. What would have been your suggestion regarding the probability of O-Ring failure using your KNN model with K=3?**

```
CREATE PROCEDURE KNN(IN Tempareture INT, IN K INT)
SELECT o_ring_failure as Failure, launch_temperature,
(SELECT COUNT(o_ring_failure)/K FROM (SELECT o_ring_failure FROM Challenger
ORDER BY ABS((SELECT Tempareture) - launch_temperature) LIMIT K) as temp
WHERE temp.o_ring_failure = Failure) AS prob
FROM Challenger
ORDER BY ABS((SELECT Tempareture)-launch_temperature)
LIMIT K;
END

CALL KNN(30, 3);
```

Failure	launch_temperature	prob
1	53	1.0000
1	57	1.0000
1	58	1.0000

IF temperature is 30 before launch, we will get 100% probability of o ring failure based on our KNN model with K = 3. Thus, we will suggest cancel the launch of space shuttle Challenger.